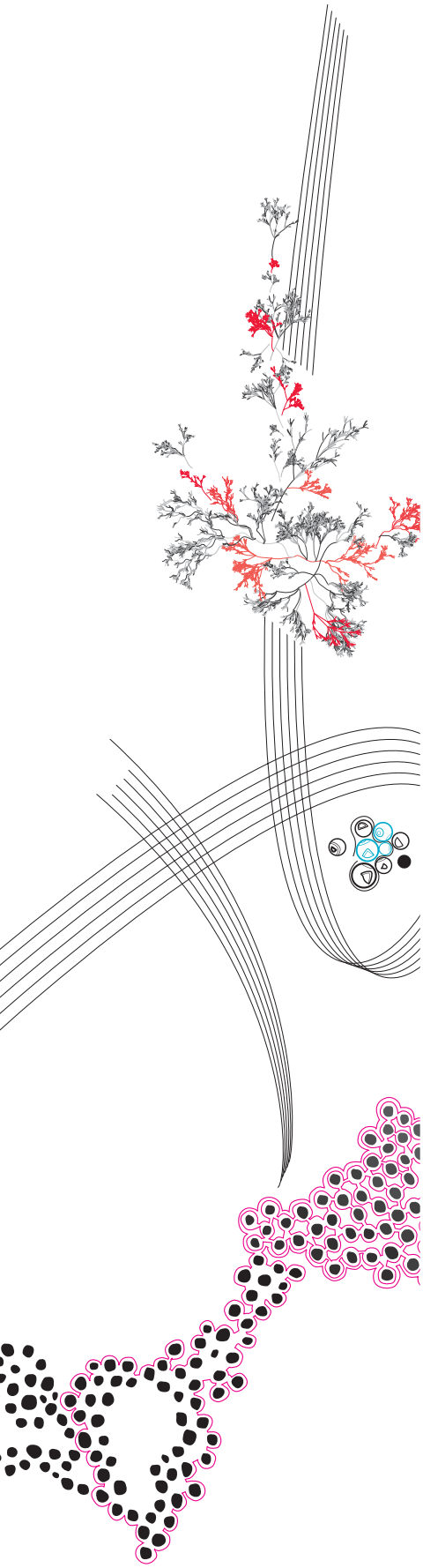


MSc Biomedical Engineering
Master's Thesis

A Real-Time Framework for Generalizable Segmentation, Tracking, and Depth Estimation using Stereo Vision

Author: Tycho Bömer

Date: May 2025

Examination committee:
dr.ir. M. Vlutters
dr.ir. A.Q.L. Keemink
dr.ing. G. Englebienne

Faculty of Engineering Technology,
Biomedical Engineering,
University of Twente

UNIVERSITY OF TWENTE.

NAKAMA ROBOTICS LAB

Abstract

Robotic systems require accurate and real-time perception to interact effectively with objects in diverse and dynamic environments. However, many existing vision pipelines struggle to generalize and fail to meet real-time constraints. This study shows that segmentation, tracking, and stereo-based depth estimation can be combined into a generalizable, real-time framework for object-level robotic interaction. The proposed framework integrates prompt-based segmentation (SAM2), a zero-shot text-based open-set detector (GroundingDINO), and stereo depth estimation using the ZED Mini stereo camera. It supports segmentation using language or spatial prompts and tracks segmented objects across frames using a modified memory mechanism. An optional co-planar refinement module, based on hybrid plane fitting using RANSAC and WLS, enhances object-level depth within segmented masks. The framework is evaluated through robotic grasping experiments across controlled scenarios, including translucent objects, low-contrast backgrounds, and cluttered scenes, at both 720p and 1080p resolutions. Quantitative results show accurate segmentation in well-contrasted scenes, reliable object tracking, and consistent depth estimation, with real-time performance maintained above 10 FPS. Performance degrades in cluttered scenes and as the number of tracked objects increases. The refinement module improves depth quality in segmented regions but introduces computational overhead and may oversimplify curved surfaces. Although generalization was demonstrated across tested conditions, all experiments were conducted in a controlled lab setting. The results confirm that the framework enables real-time object-level visual feedback for robotic manipulation under constrained conditions, demonstrating potential for deployment in dynamic environments.

Keywords: Real-time, Stereo vision, Depth estimation, Segmentation, Tracking, Generalizability

Contents

Abbreviations

BIoU Boundary Intersection over Union.

CNNs Convolutional Neural Networks.

DoF Degree of Freedom.

FCNs Fully Convolutional Networks.

FLOPs Floating Point Operations per second.

FPS frames per second.

IoU Intersection Over Union.

MAE Masked Autoencoder.

NLP Natural Language Processing.

NMS Non-Maximum suppression.

RANSAC Random Sample Consensus.

RMSE Root Mean Square Error.

SAM Segment Anything Model.

SAM2 Segment Anything Model 2.

ViTs Vision Transformers.

VOS Video Object Segmentation.

VOT Video Object Tracking.

WSLE Weighted Least Squares Estimation.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research introduction

In robotic systems, visual sensing enables robots to perceive their surroundings, recognize and localize objects, and understand the spatial layout of the environment. A key aspect of this perception is accurate depth estimation, which enables robots to perform fine-grained tasks such as grasping, navigation, and obstacle avoidance when combined with object detection, shape recognition, and tracking. To interact effectively with their surroundings, robotic systems must process this visual information in real-time [1]. Achieving these capabilities requires developments in electronics, mechanics, and artificial intelligence [2]. These developments enable robots to operate in dynamic environments and perform tasks that can go beyond traditional predefined tasks.

Dynamic environments refer to robot surroundings where conditions can often change. These settings may include moving objects, unstructured layouts, various objects requiring interaction, or changing deployment locations. The robotic system must be able to adapt to environments that differ from traditional static scenarios, where conditions remain unchanged after initial setup. Vision sensors provide rich and detailed information about the environment without prior knowledge of the deployment environment [3, 4]. Adaptability to changing conditions is particularly critical in scenarios involving human-robot interaction and collaboration, such as action and gesture recognition, robot movement in human spaces, object handover and collaborative actions, social communication, and learning from demonstration [4].

Depth information can be obtained using a variety of sensors. One of the most popular sensors is LiDAR, which uses laser technology to measure distances. However, the use of LiDAR as a depth sensor for robotic systems can be limited. One major limitation is its sparse point density, which increases with object distance [5,6]. As a result, depth completion techniques are required to generate the dense depth maps needed for precise robotic tasks [6, 7]. Furthermore, LiDAR sensors do not provide color information, which can be used as an additional cue to understand the scene, and reliable sensors remain expensive, restricting their accessibility and adoption. Like cameras, LiDAR performance also degrades under challenging environmental conditions such as sunlight, rain, or fog [8].

Advancements in deep learning have led to a growing focus on image-based methods using Convolutional Neural Networks (CNNs), which have demonstrated strong performance in stereovision-based depth estimation [9–12]. Small cameras offer an affordable and space-saving option for deployment in (compact) robotic systems. However, these methods are not without challenges. Matching pixels between stereo images to establish correspondences is computationally intensive, especially in scenes lacking distinctive visual features. Additionally, parts of the objects

can be occluded in one camera view but not in the other, complicating the matching process.

However, depth information alone is often insufficient for robot interaction with specific objects. Robots must be able to distinguish target objects from other surrounding elements. Although object detection techniques provide useful bounding boxes, they often lack the precision required for accurate robotic interaction tasks, especially in cluttered or complex scenes [13]. A more detailed representation can be achieved using image segmentation, which divides the scene into discrete groups of pixels called image segments. These segments provide detailed pixel-level information about the shape, boundary, and location of objects, which is crucial for precise robotic manipulation [14–17].

The deployment of segmentation models in robotics poses multiple challenges. First, deep learning segmentation models typically require more computational resources, limiting their applicability in real-time robotic systems. In addition, segmentation models are often trained for specific tasks or environments, limiting generalization to dynamic environments. In such environments, segmentation models must be able to perform zero/few-shot segmentation, which means they can segment the scene with little to no additional labeled data required [18]. Recent advances in prompt-based segmentation models offer a solution to the generalization problem. These types of segmentation models leverage foundation models pre-trained on large and diverse datasets, capable of performing a wide range of segmentation tasks without task-specific retraining [19, 20]. Instead of retraining all the model parameters for each new task, the models are guided by visual or textual prompts to achieve strong zero/few-shot performance without requiring additional labeled data.

Segmenting entire scenes for every frame is computationally expensive. An alternative approach involves tracking specific objects over multiple frames without re-segmenting each frame. This process involves two tasks: Video Object Segmentation (VOS) and Video Object Tracking (VOT) [21]. The VOS task focuses on extending segmentation across multiple frames and creating precise object masks. The VOT task aims to follow the location of objects over time without losing their location. The combination of both tasks enables robust and accurate segmentation tracking.

Real-time responsiveness ensures that robotic systems can react to their visually observed environment while maintaining reliability in depth estimation, segmentation, and tracking tasks. Recent advances in deep learning have increased computational complexity. Nevertheless, robotic systems must still achieve low-latency visual processing. Therefore, it is essential to balance the frame rate and the precision of the model to maintain responsibility and reliability.

1.2 Problem formulation

Without robust visual sensing, robotic systems must rely solely on non-visual sensor feedback. This severely limits their situational awareness and adaptability, especially in dynamic or unstructured environments. This study explores the feasibility of designing a framework that integrates stereovision-based depth estimation, pixel-level segmentation, and object tracking techniques into a real-time visual sensing system designed to generalize to different environments. Although each component is well established, combining them into a single pipeline that operates accurately, generalizable, and in real-time on a live camera stream has not, to the author's knowledge, been developed. Current approaches either lack real-time performance, fail to generalize, or rely on specific training data. These limitations prevent robots from interacting effectively with different environments. To evaluate the feasibility of the framework, this study uses robotic grasping as a concrete test case. This downstream robotic interaction task requires precise and timely perception of object shape, location, and depth, especially when interacting with previously unseen objects. It is a representative challenge to assess whether the frame-

work can support real-time, generalizable robotic manipulation. The design of the framework is structured around the following objectives:

1. **Stereovision-based depth estimation**: Providing the precise distance information required for object manipulation. The focus is on estimating the depth of relevant objects within the scene rather than full-scene depth estimation.
2. **Pixel-level segmentation**: Delivering detailed scene understanding at the pixel level by providing precise object masks and boundary information.
3. **Object tracking during live camera stream:** Maintaining object identity and segmentation over time, using only past and current frames.
4. **Real-time responsiveness**: Achieving frame rates that enable the system to respond in real-time to the environment. A minimum framerate of at least 10 frames per second (FPS) is considered sufficient to perform tasks such as robotic grasping.
5. **Generalizable performance**: Ensuring the framework can generalize to different environments and/or objects by achieving zero-shot/few-shot performance. This means the entire framework can generalize to unseen environments and objects without requiring additional steps.

1.3 Contributions and research question

This study contributes to the field of robot interaction based on vision by developing and analyzing a framework for real-time generalizable visual feedback using simultaneous depth estimation, segmentation, and tracking. By designing the method to work in dynamic environments, the intended goal is to generalize across different objects and environments. This makes the method suitable for a wide range of robotic applications. Although the method will be tested on the Franka Emika robotic arm at the Nakama Robotics Lab, University of Twente, the lab serves as a controlled validation setting. The research output is intended for a broader audience, including those interested in generalizable vision-based robot interaction, learning from demonstration, and mobile robot deployment. The work of this study does not attempt to optimize the architecture for specific stated design constraints but instead evaluates feasibility under realistic operational conditions.

This study addresses the following main research question:

- What design and integration strategies are feasible for enabling real-time segmentation, tracking, and depth estimation in a stereo vision framework that generalizes across environments for object-level robotic interaction?

To support this, the following subgoals are defined:

- Assess whether the framework pipeline produces reliable segmentation and depth output to perform a downstream robotic grasping task, without environment-specific tuning.
- Identify design strategies that enable segmentation-based object tracking using a live camera stream while maintaining real-time performance.
- Explore how image resolution and the number of tracked objects affect the real-time performance of the framework.

Chapter 2

Background

2.1 Stereo depth estimation

2.1.1 Stereo depth estimation problem

A stereo camera captures two images of the same scene from slightly different viewpoints, enabling the estimation of absolute depth, which refers to the actual distance from the camera to objects in the scene [22, 23]. By matching corresponding points in both images, it is possible to compute the disparity. Disparity is the difference in the horizontal pixel positions of the same point across the two stereo images. Triangulation estimates object depth by leveraging the known geometry of the stereo camera setup and the computed disparity. The general simplified stereo depth estimation problem is shown in Figure 2.1.

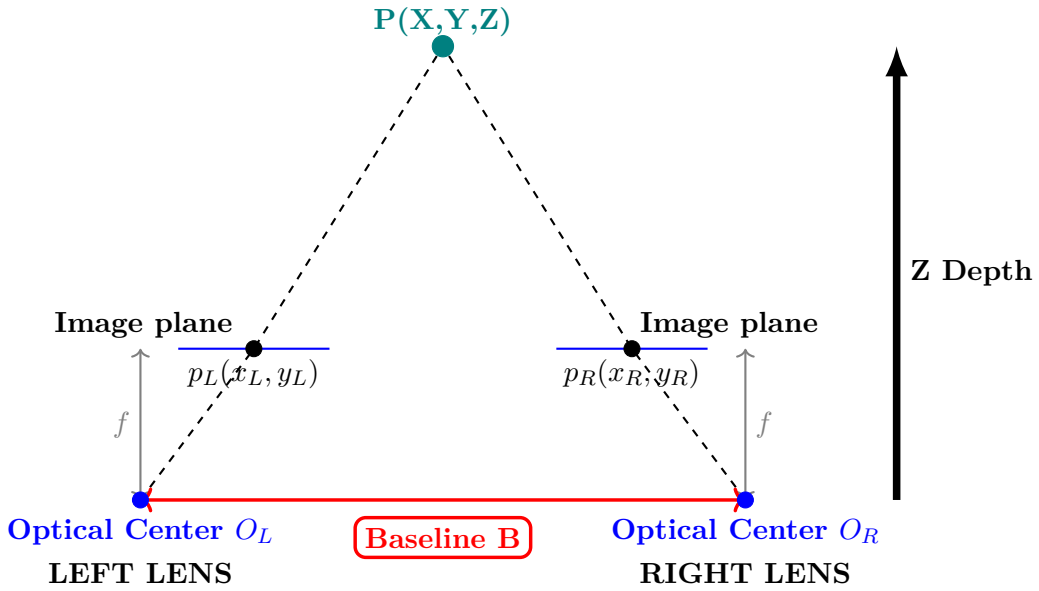


Figure 2.1: *Schematic illustration of the stereo depth estimation problem. The diagram represents two camera viewpoints separated by baseline B, where a scene point $P(X, Y, Z)$ is projected onto both image planes at $P_L(x_L, y_L)$ and $P_R(x_R, x_L)$. The disparity is defined as the difference in pixels between projected coordinates, and the depth Z is computed using the inverse relationship with disparity.*

Two corresponding image points are identified in the left and right camera views, forming equivalent triangles. The relationship between disparity and depth is given by:

$$\text{disparity} = x_L - x_R = \frac{Bf}{Z} \quad \Rightarrow \quad Z = \frac{Bf}{x_L - x_R}. \tag{2.1}$$

Where:

$x_L - x_R$: Disparity, the pixel shift between the corresponding points in the image plane,
B : Baseline, the distance between the two cameras,
f : Focal length of the camera,
Z : Depth, the distance of the point from the camera.

The depth of a point in a scene is inversely proportional to the disparity between its projections on the left and right image planes. Points closer to the camera appear farther apart in the two images, while distant points appear closer together. Using this principle, the depth map can be calculated for individual pixels in the scene. The stereo depth estimation problem can be addressed using various methods. Early methods rely on traditional stereo-matching algorithms, while recent ones use deep learning-based approaches.

2.1.2 Traditional methods

Stereo depth estimation is based on the concept of epipolar geometry. When the optical axes of both lenses are aligned horizontally, which means that the lenses are at the same height, the search for corresponding points can be constrained to a single horizontal line, known as the epipolar line. By restricting the search space to this one-dimensional axis, the algorithm's computational complexity is reduced, transforming the multidimensional problem of stereo-matching into a one-dimensional problem. Traditional stereo matching algorithms are typically classified into three categories: local matching, global matching, and semi-global matching, according to the pixel range they process [24, 25].

- **Local Matching:** In local matching, a small neighborhood is compared around each pixel in one image with the corresponding region in the other. The most similar region is identified, and its center is used as the matching point. Although local matching is computationally efficient, it may lack high accuracy, particularly in scenes with ambiguous data.
- **Global matching:** Global matching, on the other hand, computes a cost map between a pair of stereo images and utilizes energy-based optimization techniques to find an optimal path in the cost map. This optimal point serves as the matching point for each pixel in the other image. Although global matching offers high accuracy, it often comes with a higher computational cost.
- **Semi-global matching:** Semi-global matching provides a balance between efficiency and accuracy by computing the cost function for each pixel and aggregating matching costs from multiple directions. This method improves depth estimation by enforcing smoothness constraints across the disparity map while maintaining reasonable computational complexity. This approach offers a good compromise between computational efficiency and accuracy.

Although traditional methods can deliver reliable results in well-structured scenes, they suffer from major limitations in complex environments. Occlusions, featureless surfaces, and reflective objects can reduce performance, as these methods do not understand object shapes and/or scene semantics [26]. To overcome these challenges, modern approaches redefine stereo depth estimation as a learning-based problem. The next section explores deep learning-based methods for depth estimation.

2.1.3 Deep learning-based methods

Deep learning-based methods often depend on large annotated stereo datasets to achieve high performance and adaptability to different environments. However, the availability of high-quality stereo datasets is very limited [27]. Most available datasets are designed for specific use cases, such as autonomous driving. This scarcity limits the training of stereo models and limits their ability to generalize to dynamic real-world environments. As a result, many learning-based approaches require extensive dataset-specific fine-tuning.

Deep learning-based methods are divided into two main categories: feature-based disparity matching and end-to-end disparity matching. The former uses a hybrid approach that integrates learned representations with traditional stereo matching, while the latter directly predicts disparity maps using fully deep networks.

Feature-Based Disparity Matching

The methods in this class mimic traditional stereo-matching techniques by explicitly learning how to match pixels in input images by minimizing an energy function formulated as [12]:

$$E(D) = \sum_{x} C(x, d_x) + \sum_{x} \sum_{y \in \mathcal{N}_x} E_s(d_x, d_y). \tag{2.2}$$

Where:

D	: Disparity map assigning a disparity d_x to each pixel x,
x, y	: Image pixels,
$\mathcal{N}_x$	: Neighborhood of pixel x,
$C(x, d_x)$	: Matching cost at pixel x for disparity d_x,
$E_s(d_x, d_y)$	: Smoothness term encouraging similar disparities between x and y,
$E(D)$	: Total energy to be minimized.

These correspondences are converted into a disparity map and then into depth at each pixel. The general outline of a feature learning architecture consists of three main modules [12, 28]:

- Feature Extraction Module: Uses CNNs or other architectures to extract features from input images.
- Feature Matching and Cost Aggregation Module: Matches features from the left and right images, creating a cost volume that represents disparity likelihoods at each pixel.
- Disparity/Depth Estimation Module: Processes the cost volume to compute a final disparity map, which is then converted to depth using known camera parameters.

In feature-based matching methods, each module is trained independently or in stages. The training optimizes each component to perform its specific function effectively and requires multiple passes through the network before minimizing the cost function. This approach ensures that features are accurately matched and that disparities are estimated with precision.

In addition to the main modules, various extensions and/or refinements have been proposed to enhance performance. Some methods incorporate refinement modules to iteratively improve the accuracy of the depth maps [28]. Others extend the approach to multiview setups, taking advantage of additional perspectives to provide a more robust depth estimation [29]. These variants offer improved precision and applicability in more complex and dynamic environments. However, deployment in real-time scenarios is challenging due to higher computational costs.

End-to-End Disparity Matching

Methods in the second class solve the stereo-matching problem using an end-to-end trainable pipeline. During training, the entire disparity estimation process is optimized using a loss function that supervises the predicted disparity map. These models benefit from an architecture that avoids separate matching and regularization modules, instead relying on joint optimization using loss functions [12,30]. Unlike feature-based methods that process local patches, end-to-end methods extract dense feature maps from the entire image pair. These feature maps maintain or reduce the spatial resolution of the input images, thereby capturing a more global context of the scene.

A crucial component in end-to-end methods is the use of cost volumes [12, 31]. Cost volumes represent the similarity between the features of pixels at different disparities by computing the feature correspondence between the left and right images. Traditional hand-made matching costs are replaced by learned cost functions, allowing deep networks to optimize the disparity estimation. End-to-end disparity matching can be approached using supervised, unsupervised, and weakly supervised learning:

- **Supervised Learning:** These methods require a large amount of ground truth disparity data for training. Models learn to predict disparity maps by minimizing the difference between predicted and ground-truth disparities. Although highly accurate, the requirement for extensive labeled data can be a limitation.
- **Unsupervised Learning:** These methods do not require ground-truth disparity data. Instead, they rely on the photometric consistency between left and right images to supervise learning. The model aims to minimize the photometric error between the reconstructed image and the original image without supervision. This approach reduces the dependency on labeled data, but may struggle with textureless regions and occlusions.
- **Weakly Supervised Learning:** These methods combine supervised and unsupervised learning elements. They may use sparse ground-truth data or weak supervisory signals (e.g., depth from other sensors, semantic information, edge maps) to guide the learning process. This approach aims to balance the accuracy of supervised learning with the data efficiency of unsupervised learning.

End-to-end trained models often produce lower-resolution disparity maps. Instead of upsampling the cost volumes, which is computationally expensive due to the high dimensionality, the predicted low-resolution disparity maps are upsampled. Lightweight 2D convolutional blocks perform this refinement by progressively reintroducing high-frequency details [12, 32, 33]. Refinement of the global context information using these blocks improves the accuracy of the disparity, while avoiding high-resolution cost-volume processing, reducing computational cost and memory usage. This strategy is particularly beneficial for real-time depth estimation.

2.2 Image Segmentation

Image segmentation involves partitioning the image into several meaningful segments. This provides the image with global information about the scene and its content. With increasing success using deep learning models in different segmentation tasks, it has been deployed in a wide range of computer vision tasks such as scene understanding, medical image analysis, robotic perception, video surveillance, augmented reality, and image compression [34]. This study has three types of segmentation methods worth mentioning: semantic segmentation, instance segmentation, and the more recent promptable segmentation.

2.2.1 Semantic segmentation

Semantic segmentation is the subclass of segmentation that deals with determining the semantic class to which each pixel of an image belongs [34,35]. It is a pixel-level labeling task within a set of (possibly different) object classes that provides a dense understanding of objects and classes in the scene (see Figure 2.2a).

Traditional methods use manually crafted image features for model training. The development of Fully Convolutional Networks (FCNs) for semantic segmentation enabled end-to-end learning by using end-to-end trainable convolutional networks [36]. The proposed architecture used an encoder-decoder structure. The encoder downsamples the input image using convolutions from a feature map with high resolution to a feature map with a lower resolution representation. The decoder performs upsampling on the compressed representation to transform the output with embeddings back to the original size. This encoder-decoder structure has the advantage of being able to learn high-level and low-level features, which are essential for accurate semantic segmentation.

Improvements have been made to the structure of FCNs. U-Net introduces skip connections between the encoder and decoder [37]. This preserves spatial details during upsampling by directly using the encoder's high-resolution features in the decoder's layers. Skip connections help recover textures lost during downsampling. This design is particularly effective for tasks that require precise localization.

The most recent development is the usage of Vision Transformers (ViTs) for semantic segmentation tasks. These architectures have a multi-head self-attention mechanism, allowing the model to capture long-range dependencies and global context of the image [38,39]. ViT-based segmentation models, such as SETR [39] and Mask2Former [40], have demonstrated the effectiveness of transformer-based architectures by combining ViTs with CNN-based designs, allowing the model to handle complex scenes with various object classes. However, a notable drawback of ViTs is their dependence on large training data [38]. To mitigate this limitation, self-supervised learning approaches have been explored, enabling ViTs to learn meaningful representations using unlabeled data [41].

2.2.2 Instance segmentation

Instance segmentation is a subclass of segmentation that determines the semantic instances of individual objects in the scene. In contrast to semantic segmentation, which assigns a class label to the pixels in the mask, instance segmentation classifies pixels and distinguishes different instances [14]. Multiple instances of the same class are segmented into unique segmented masks (see Figure 2.2b), enabling object separation.

With the adaptation of FCNs for instance segmentation tasks the Mask R-CNN network was proposed [42]. This network builds upon the Faster-RCNN object detection network by adding a segmentation network. A Region Proposal Network is used to generate candidate regions that are likely to contain objects, which are then refined and classified, before predicting the pixel-level masks. Since instance segmentation requires more complex output predictions compared to semantic segmentation and object detection, real-time applications have been limited. The first major model that addresses this limitation is YOLOACT [43]. It adopted a fully convolutional topology that ran two tasks in parallel: mask prototype generation and mask coefficient estimation, increasing inference speed while maintaining accuracy for the segmentation tasks.

Another different approach to instance segmentation has been made in SOLO (Segmenting Objects by Locations) [44]. SOLO reformulates instance segmentation as a category-aware pixel localization problem, diverging from the region proposal-based method first used in Mask R-CNN. The model decouples the instance segmentation task into two parallel branches: a category

branch and a mask branch. The category branch predicts the semantic class in each grid cell, while the mask branch generates the corresponding instance mask. Since SOLO has location-aware information, complex post-processing steps such as Non-Maximum suppression (NMS) are eliminated, increasing computational efficiency.

Instance segmentation also adopted the usage of ViTs. Transformer-based object detection models, such as DETR, have been extended with an additional mask prediction head attached to the output of the decoder, enabling instance-level mask generation [40, 45]. The self-attention mechanism of ViTs can capture global context and long-range dependencies. By formulating the problem as a set prediction problem, the need for hand-designed components such as anchor boxes and NMS is eliminated. This allows for an end-to-end training approach, first introduced in the ISTR model [46]. This training method enables the model to predict low-dimensional masks and match them with ground truth mask embeddings, resulting in a set-based loss.

More recent advances in instance segmentation focus on multi-task learning objectives by developing hybrid structures. Swin Transformer [47] combines FCNs and ViTs with shifted windows, achieving dense global and local features. This information maintains versatility, making it usable for a broader range of vision tasks, including instance segmentation, image classification, and object detection. In addition to fully supervised end-to-end trainable methods, recent techniques use self-supervised and semi-supervised learning to address the issues of requiring large datasets for training vision transformers. Adopting these training techniques allows for better generalization outside of the training dataset.

(a) *Semantic segmentation* **(b)** *Instance segmentation*

Figure 2.2: *Example distinction: Semantic segmentation (a) and Instance segmentation (b) [48].*

2.2.3 Promptable segmentation

Promptable segmentation builds on the concept of prompt-based learning and tuning, originally developed in natural language processing (Natural Language Processing (NLP)). This technique has recently been extended to ViTs [19, 20]. It enables large foundation models to adapt their pre-trained knowledge to downstream segmentation tasks using prompts as guidance. The idea behind prompt-based learning in foundation models is that the pre-trained visual backbone remains frozen, while only a small set of tunable parameters is optimized. By keeping the visual foundation model frozen during training, segmentation can be dynamically guided by task-specific prompts, allowing flexibility without retraining the entire model. This approach improves generalization for diverse tasks and datasets. Prompts can take on various forms, such as points, boxes, text, or even other images, to guide the segmentation process.

Key breakthroughs in promptable segmentation are the Segment Anything Model (SAM) [49] and its successor Segment Anything Model 2 (SAM2) [50]. The following two sections outline these two segmentation models in greater detail.

Segment Anything Model (SAM)

SAM [49] is a foundation model for zero-shot segmentation that generates output masks based on user-defined prompts. It supports four types of input prompts: points, bounding boxes, masks, or free-form text. There are three main components in the SAM model, as illustrated in Figure 2.3: a large image encoder, a flexible prompt encoder, and a fast mask decoder.

Unlike semantic segmentation or instance segmentation, SAM is designed to be class-agnostic. This means that it does not associate segmentation masks with predefined object categories. Instead, it produces binary masks that segment objects based solely on the provided prompts, such as points, boxes, or masks. Although it is possible to use free-form text prompts through CLIP (Contrastive Language-Image Pretraining) [51], this introduces only a limited form of semantic understanding. As noted by the authors, the text-to-mask capability is exploratory and not robust in most cases. Although users can describe objects using natural language, the model output remains a binary mask without an attached semantic label.

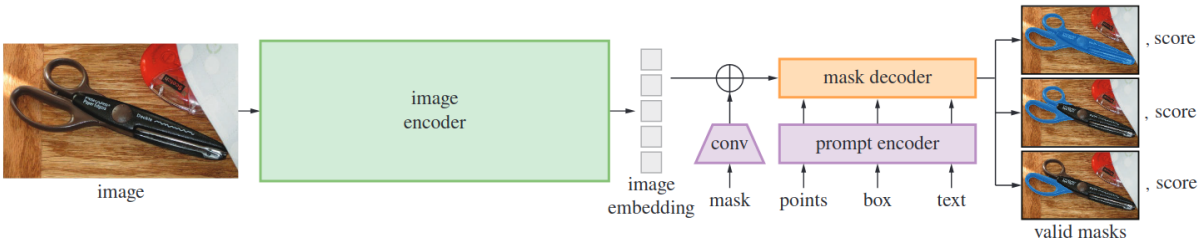


Figure 2.3: *Overview of Segment Anything Model (SAM). The larger image encoder produces lower-resolution image embeddings. The image embedding can be directly combined with the mask points box or text prompts by the mask decoder to output predicted segmentation masks [49].*

Image Encoder: A Masked Autoencoder (MAE) [52] pre-trained ViT serves as the backbone for processing images into a generalized feature representation. Specifically, SAM uses the ViT-Huge model using 16x16 patch embeddings, which provides a strong balance between computational efficiency and detailed feature extraction. The MAE pretraining allows the encoder to learn robust feature representations by reconstructing missing image patches during training. A larger ViT has been selected as an image encoder to extract a higher-dimensional feature representation, capturing both fine-grained details and global context. The higher number of Floating Point Operations per second (FLOPs) associated with larger image encoders increases the computational load. Since the image encoder runs once per image, some of the load is shifted to the lightweight mask decoder.

Prompt Encoder: For the three available spatial prompts (points, boxes, and masks), SAM employs a learnable embedding approach, where the input coordinates are projected into a high-dimensional space. These embeddings act as query representations, guiding the model to focus on the relevant image features extracted by the encoder. For free-form text prompts, SAM uses CLIP [51] to connect text descriptions with visual features. The trained CLIP text encoder understands the meaning of words and can match them with objects in the image. This allows SAM to process text-based queries and generate object masks using user-input text in natural language.

Mask Decoder: The mask decoder is responsible for generating the final segmentation masks by combining image features from the encoder with embedded prompts from the prompt encoder. These inputs are processed alongside an output token, leveraging self-attention and cross-attention mechanisms to refine the output. The attention mechanisms operate in both the image-to-prompt and prompt-to-image directions, ensuring that all embeddings interact and are updated correctly. The final prediction is made using a multilayer perceptron as a dynamic linear classifier. This results in a total of three possible predictions, of which the most confident is chosen.

SAM was trained on the SA-1B dataset[1], one of the largest segmentation datasets to date, containing 11 million images and 1.1 billion segmentation masks. This large-scale data set was designed to improve generalization between various objects, scenes, and lighting conditions, allowing zero-shot performance in segmentation tasks.

Although SAM can process the prompts in real time, the overall performance of the model does not meet the real-time requirements when using large image encoders, as stated as an important limitation in the paper. The ViT-Huge backbone in SAM is highly effective in extracting detailed feature representations, but this comes at the cost of high GPU memory usage, increased inference latency, and increased number of floating-point operations. Several adaptations have been made to the original model that focus on reducing computational costs while maintaining accuracy to address these limitations. These include EfficientSAM [53], MobileSAM [54], TinySAM [55], EdgeSAM [56] and FastSAM [57].

In addition to computational constraints, SAM lacks tracking capabilities in its design, making it unsuitable for VOS and VOT without requiring additional models. SAM generates independent segmentation masks per frame without tracking the masks across multiple frames. External tracking mechanisms have been used in combination with SAM to make tracking tasks possible:

- XMem [58] maintains a memory bank with past features, allowing for consistent segmentation between multiple frames. However, using XMEM introduces additional latency as a result of feature conversion, storage, and retrieval overhead.
- Cutie [59] uses feature aggregation and efficient memory updates. However, it still relies on an external memory network, which increases computational costs.

Segment Anything Model 2 (SAM2)

SAM2 [50] is the improved foundation model for prompt-based segmentation, building on the original SAM while introducing major improvements in efficiency and implementing video tracking using memory integration. Using a memory bank and the memory attention mechanism enables the tracking of objects across frames and reduces the required computational costs, making it ideal for real-time tasks VOS and VOT. The updated architecture of SAM2 is shown in Figure 2.4. SAM2 introduces several key architectural components: a refined image encoder, an enhanced prompt encoder, a memory-aware mask decoder, a memory bank, and a memory-attention mechanism.

[1]SA-1B dataset Information: `https://ai.meta.com/datasets/segment-anything/`

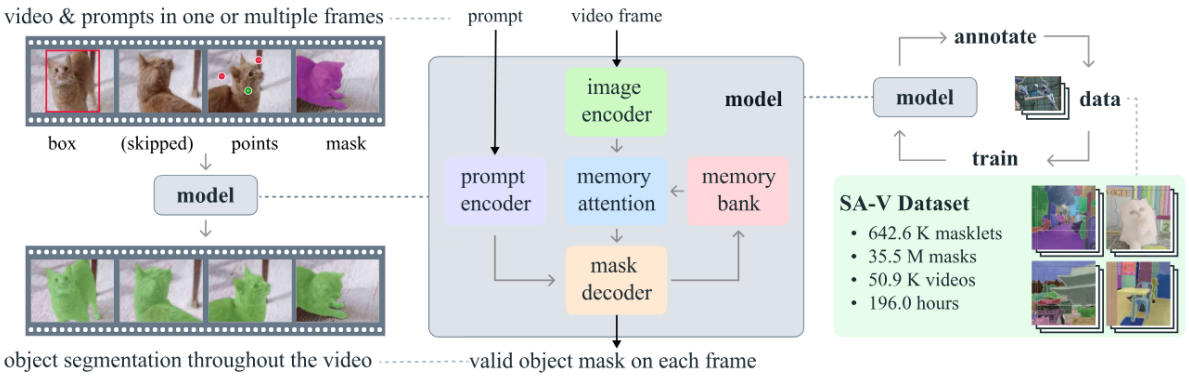


Figure 2.4: *Overview of Segment Anything Model 2 (SAM2): prompt-based segmentation using boxes, points, or masks (left), the SAM2 foundation model with image encoder, prompt encoder, mask decoder, and memory bank (middle), and the SA-V training dataset (right) [50].*

Image Encoder: The image encoder has been upgraded to Hiera ViT [60]. The Hiera encoder is pre-trained using the strong visual pretext MAE. This architecture simplifies multi-stage vision transformers by organizing them hierarchically. The approach suggests that spatial biases required for vision tasks can be learned through proper pertaining instead of requiring architectural complexity.

Prompt Encoder: The prompt encoder in SAM2 follows the same design as in SAM. The integrated vision model, CLIP, has been removed to allow better optimization and compilation of spatial prompts (box, point(s), mask), resulting in more efficient inference. Additionally, the text-to-mask functionality provided by CLIP in SAM was reported to be exploratory and not robust in most cases.

Memory encoder: The memory encoder processes the segmentation mask from the mask decoder and encodes it in a small memory embedding. This is achieved by downsampling the mask by using a lightweight convolutional module and fusing it with the frame embedding from the image encoder. These embeddings are passed through additional convolutional layers before being stored in the memory bank.

Memory bank: The memory bank is one of the key innovations implemented in SAM2. It uses a FIFO queue that retains segmentation information from already segmented frames, which enables more efficient segmentation and tracking. Feature maps from the last N unprompted frames are stored along with segmentation information from M prompted frames. This ensures that the model has both information on recent tracking history and reference prompt(s). Additionally, the memory bank integrates lightweight object pointers derived from the mask decoder's output to capture high-level semantic information about the segmented object(s).

Memory attention: The memory attention mechanism allows SAM2 to associate objects between frames by attending stored representations in the memory bank. This mechanism refines the predictions by taking advantage of the temporal context. For video tracking, the mask prediction from the previous frame is used as a reference in conjunction with memory attention. Self-attention is used for the current frame, while cross-attention is used between the previous frame's information stored in the memory bank.

Mask Decoder: The memory-aware mask decoder refines the segmentation output by combining spatial and temporal information. The mask decoder in SAM2 is optimized to integrate prompt information with memory-based attention.

SAM2 is trained on the Segment Anything Video (SA-V) dataset[2]. The dataset provides high-quality segmentation annotations created in video sequences and images. This ensures that models can learn and handle temporal variations, occlusions, and complex object interactions. As a result, SAM2 supports zero-shot segmentation capabilities, which is essential for generalizing across different environments without requiring task-specific retraining [50].

SAM2 effectively isolates objects in simple scenes where salient objects stand out clearly from the background [61]. The model achieves high segmentation accuracy in diverse environments, and it generalizes well to unseen object categories. However, in more challenging environments, such as cluttered scenes, occlusions, or instances where objects blend into their surroundings, the ability to generalize becomes less reliable. In these cases, the model struggles to define precise boundaries, which is particularly limiting for tasks that require high-resolution fine-grained segmentation.

The integrated SAM2 tracking mechanism eliminates the need for external tracking models. Unlike SAM, which relied on secondary models such as Cutie or XMEM, to enable object tracking. In addition, the integrated memory mechanism of SAM2 outperforms Cutie and XMEM [50]. This makes SAM2 the preferred choice for real-time applications.

2.3 Evaluation metrics

Several quantifiable metrics can be used to evaluate segmentation accuracy, depth estimation accuracy, and performance.

Depth Accuracy

When ground truth depth information is unavailable or difficult to obtain, the reliability of a depth estimation method can be assessed using 3D geometric references. One of these approaches involves using fiducial markers with known sizes and easily detectable patterns, such as ArUco markers [62], as reference points. AruCo markers allow for 6 Degree of Freedom (DoF) pose estimation, providing position (X,Y,Z) and orientation $(\theta_x, \theta_y, \theta_z)$ relative to the camera lens, following the standard camera projection model between camera coordinates and world coordinates [63] (see Figure 2.5).

This relation is mathematically described by the Perspective-n-Point (PnP) projection model, which requires at least $n \geq 3$ projected points to be solvable [64]. This model is expressed as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R \mid t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.3}$$

Where:

(u, v)	: 2D pixel coordinates in the image,
K	: Camera intrinsic matrix,
R	: 3x3 rotation matrix (extrinsic parameters)
t	: 3x1 translation vector (extrinsic parameters),
(X, Y, Z)	: 3D world coordinates of the marker (with Z as the depth).

Calibrating the camera incorporates intrinsic parameters like focal length and distortion coefficients into the markers 3D pose estimation.

[2]SA-V dataset information: `https://ai.meta.com/datasets/segment-anything-video/`

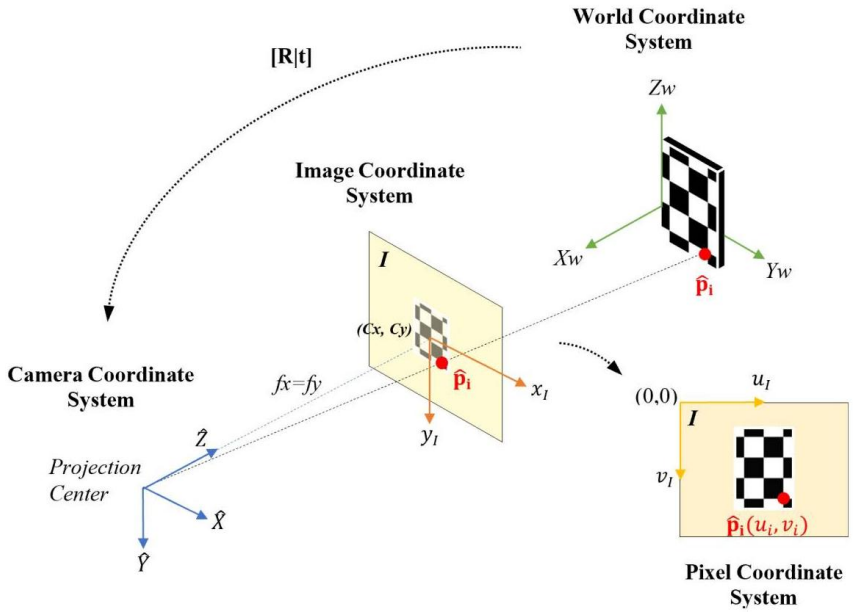


Figure 2.5: *Illustration of the camera projection model for depth estimation. The figure shows the relationship between the 3D world coordinates of a marker and its projected 2D coordinates in the camera frame [63].*

Instead of evaluating the error only along the estimated depth axis (Z-axis) in the image coordinate frame, the evaluation projects the estimated points into 3D positions within the camera coordinate frame to form point clouds. The estimated sensor depth values and the ArUco-derived ground truth values are projected. The error is then evaluated in all three directions (X,Y,Z), which is a more complete error metric. The Euclidean error between the estimated 3D points and derived ArUco 3D points is computed as:

$$e_i^2 = \left\| P_i - \widehat{P}_i \right\|^2 = \left(X_i - \widehat{X}_i \right)^2 + \left(Y_i - \widehat{Y}_i \right)^2 + \left(Z_i - \widehat{Z}_i \right)^2. \tag{2.4}$$

Where:

P_i	: 3D position estimated by the depth model in the camera coordinate frame,
$\widehat{P}_i$	: 3D position derived from ArUco markers in the camera coordinate frame,
(X_i, Y_i, Z_i)	: Estimated coordinates of the point,
$(\widehat{X}_i, \widehat{Y}_i, \widehat{Z}_i)$	: Ground truth coordinates of the same point,
e_i	: Euclidean error between the estimated and ground truth 3D positions.

To analyze depth error behavior, the evaluation groups 3D errors into depth distance bins based on the derived distance Z^j between the ArUco marker and the camera. For each bin centered around Z^j, the evaluation computes the RMSE using N_j points assigned to that bin, where N_j denotes the number of samples in bin j.

$$\text{RMSE}\left(Z^j\right) = \sqrt{\frac{1}{N_j} \sum_{i=1}^{N_j} e_i^2}. \tag{2.5}$$

Following [63], the evaluation models the Root Mean Square Error (RMSE) values using curve fitting to describe the depth error characteristics. It applies both a second-order polynomial and an exponential function to approximate how the error increases with distance.

$$E_{\text{poly}}(Z) = aZ^2 + bZ + c \tag{2.6a}$$
$$E_{\text{exp}}(Z) = ae^{bZ}. \tag{2.6b}$$

The exponential model includes two parameters, a and b, while the polynomial model includes three: a, b, and c.

Segmentation accuracy

The segmentation evaluation compares the predicted segmentation masks to ground truth annotations using two commonly used metrics: Intersection Over Union (IoU) and Boundary Intersection over Union (BIoU).

IoU: A widely used segmentation metric that measures the overlap between a predicted mask and its corresponding ground truth. It is defined as the ratio of the intersection area to the union area:

$$IoU = \frac{|(G \cap P)|}{|(G \cup P)|}. \tag{2.7}$$

Where:

G : Ground truth mask,
P : Predicted mask.

BIoU: Although IoU is a reliable metric for evaluating overall segmentation accuracy, it tends to introduce a bias towards larger objects. This is because larger objects contribute more to the total union area, making errors in smaller objects less significant on the overall IoU score [65]. BIoU combines the evaluation of the overall mask and the boundary regions. It is more sensitive to errors near the edges of the object while considering the rest of the mask (see Figure 2.6). BIoU can be calculated using the following equation:

$$\text{Boundary IoU} = \frac{|(G_d \cap G) \cap (P_d \cap P)|}{|(G_d \cap G) \cup (P_d \cap P)|}. \tag{2.8}$$

Where:

G : Ground truth mask,
P : Predicted mask,
G_d : Boundary region of the ground truth mask, dilated by d,
P_d : Boundary region of the predicted mask, dilated by d.

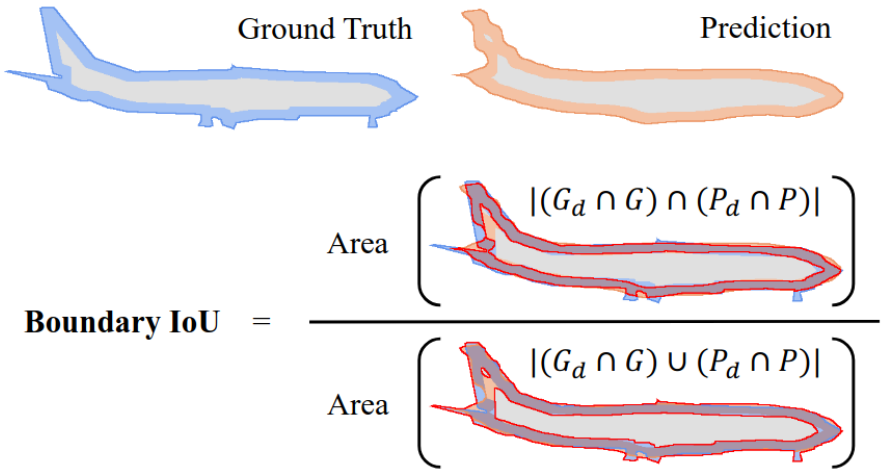


Figure 2.6: *Illustration of Boundary IoU computation. (Numerator) The intersection between the boundary regions of the ground truth and predicted masks, obtained by dilating both masks by a distance d. (Denominator) The corresponding union of these boundary regions. The ratio of these areas is the BIoU [65].*

Real-time performance

The performance of a workflow can be evaluated by measuring the speed of execution. This can be expressed as the amount of FPS the system can process.

Chapter 3

Methods

3.1 Design considerations

The development of the proposed visual perception framework follows a feasibility-driven design process. The aim is to integrate well-performing components into a modular pipeline that satisfies the design objectives stated in Section 1.2. These objectives include: providing stereovision-based depth estimation for objects, delivering pixel-level segmentation masks for scene understanding, object tracking to maintain identity and segmentation on live camera streams, achieving a real-time processing rate of at least 10 FPS, and enabling generalization to new objects and environments without environment-specific tuning or retraining. This section discusses the considerations for framework components. The technical implementation details are discussed in the next section.

Depth estimation using ZED Mini

The internal depth sensing module of the ZED Mini stereo camera was selected to provide dense, real-time depth maps. This module uses a deep stereo-matching network trained specifically on stereo image pairs captured with ZED cameras. As discussed in Section 2.1.3, the availability of high-quality stereo vision datasets is limited, and most are tailored to specific use cases such as autonomous driving. These limitations make it difficult to train stereo models that generalize well across diverse, dynamic environments. In contrast, the ZED depth model performs reliably across a range of settings without any retraining, offering zero-shot stereo depth estimation. Since the model runs natively as a compiled, CUDA-accelerated implementation, it does not introduce additional latency beyond image acquisition. However, because the model is closed-source, only the final depth map is accessible. Intermediate calculations, such as feature extraction and disparity, are not exposed. Still, this combination of strong generalization and real-time performance addresses both the latency and generalization challenges identified in this study.

Segmentation and tracking with SAM2

Prompt-based segmentation is selected for its strong generalization capabilities. Foundation models are pre-trained on large and diverse datasets, allowing segmentation for a wide range of tasks without specific retraining [19,20]. SAM2 extends promptable segmentation by embedding object memory directly in its architecture, allowing tracking between frames without relying on separate tracking models. This architectural design simplifies integration and avoids the latency overhead introduced by adding external tracking models. While the current release supports both single-image and video-based segmentation, the architecture must be converted

to work with live camera streams, where future frames are not available. The strong zero-shot performance and tracking capabilities make SAM2 well-suited to meet the design objectives of this study.

Text-based detection with Grounding DINO

GroundingDINO [66] is used to reintroduce text-based control while remaining compatible with the spatial prompt encoder of SAM2. Instead of relying on fixed class labels, the model enables open-set zero-shot detection through natural language. Using language as prompts aligns well with the prompt-based design of SAM2. Although alternative open-set detectors were considered, GroundingDINO was ultimately selected because of its strong zero-shot performance and its ability to generate precise bounding boxes.

3.2 Framework design

This section provides an overview of the proposed method, which integrates segmentation based on SAM2 and depth estimation using the Stereolabs ZED-Mini stereo camera. The workflow consists of multiple interconnected components that process stereo image data to generate object segmentation masks and the corresponding depth information (see Figure 3.1).

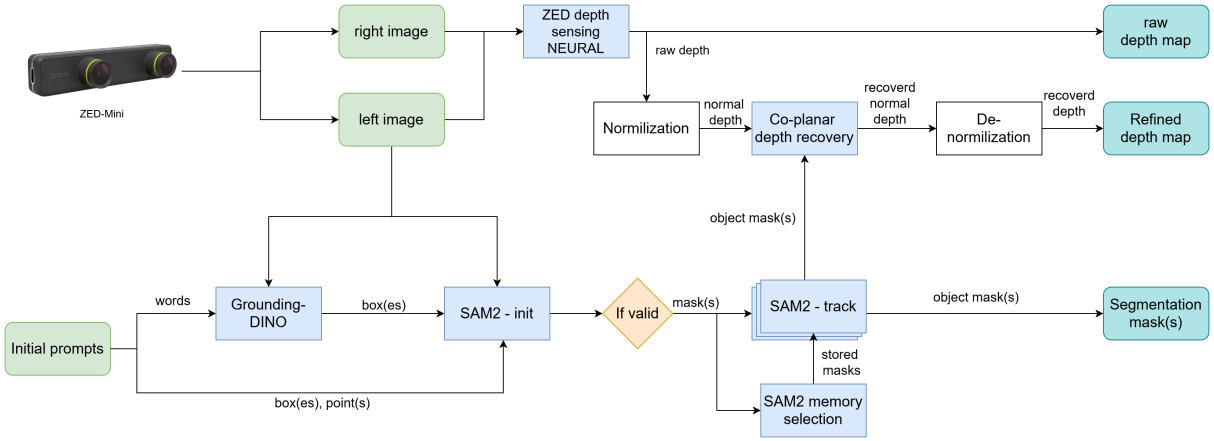


Figure 3.1: *Illustrative overview of the designed framework.*

The proposed method begins by capturing a stereo image pair using the ZED Mini camera. The left image is used as input for segmentation, while the right image is used solely for computing depth. Segmentation is initialized through user-defined prompts, which can be spatial (bounding boxes or points) or textual. Spatial prompts are passed directly to SAM2, while textual prompts are first processed by GroundingDINO [66], a language-guided open-set object detector that returns bounding boxes corresponding to the described objects. These bounding boxes are then used to initialize segmentation in SAM2. During tracking, segmentation masks from the previous frame are reused as prompts in subsequent frames. The predicted segmentation masks are passed to a memory management module, which handles the tracking between frames by selectively updating the memory based on the valid segmentation masks. In parallel, the ZED's NEURAL depth sensing module extracts depth information from the stereo image pair. Optionally, the resulting raw depth map can be refined by combining it with the segmentation output in a co-planar depth recovery module. This module aims to improve depth quality within object mask regions.

3.2.1 SAM2 adaptations

Since the SAM2 model has already been discussed in Section 2.2.3 with its most important components highlighted, it will not be repeated here. Instead, only important adaptations made to the model architecture are discussed. In addition, a highly detailed workflow of the SAM2 model has been created and is provided in Appendix B. This workflow is directly derived from the original SAM2 paper [50] and the associated source code. It is included in the Appendix rather than in this section to maintain focus on adaptations made to the model instead of repeating the official paper.

Memory management

The framework introduces a dynamic memory management method to adapt the memory network of SAM2 for a real-time camera stream. The framework initializes segmentation in the first frame and is dynamically updated as new frames arrive. Memory is only updated when a valid segmentation result is available. The system avoids storing empty or low-confidence masks when an object is occluded or absent, preventing past valid states from being overwritten with irrelevant data. As a result, even after long occlusions, the memory of that object can be used as a reference to re-identify the objects based on similarity to the last valid memory entries. The original memory selection mechanism used a fixed window that slides over embedded memory frames in the memory bank, relying on cross-attention across both past and future frames. The redesigned mechanism removes the dependency on future frames, enabling compatibility with streaming input. It performs cross-attention only between the current frame and valid past frames. The difference between original memory selection and new memory selection is shown in Figure 3.2.

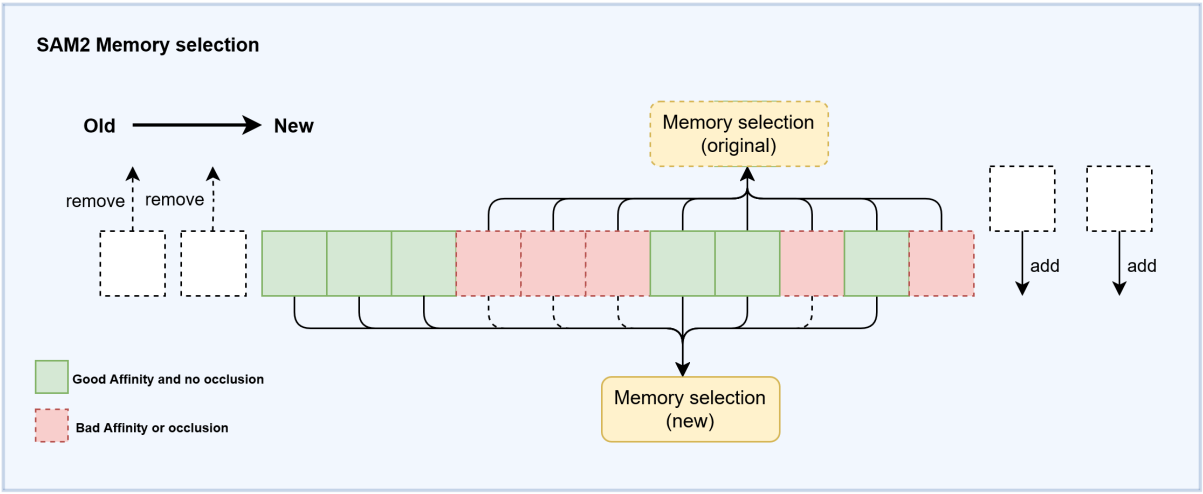


Figure 3.2: *Schematic overview of the original memory selection (upper part) compared to the new memory selection (lower part). Older frames are removed to keep memory efficient for better real-time capabilities. Frames with no valid information (low-affinity score or occluded) are not used in the new memory selection mechanism.*

Text prompt integration

GroundingDINO enables text-based input support, which is not handled by the prompt encoder of SAM2. This transformer-based object detector performs open-set object detection by combining visual and textual information. The model extracts dense image features using a Swin Transformer [47] as its visual backbone, while a BERT-based encoder [67] processes the user-provided text prompt to produce a contextual text representation. These two modalities

are combined in a cross-modality decoder. This decoder performs self-attention on the image features, self-attention on the text features, and cross-attention between image and text. This fusion allows the model to localize regions in the image that correspond semantically to the input text. GroundingDINO does not predict semantic class labels or segmentation masks. Instead, it predicts bounding boxes for regions that match the text description. These boxes are used as prompt(s) to SAM2, which predicts segmentation masks for the object(s) defined by the text prompt.

Tracking

Object tracking is initialized in one of two ways:

1. Using predefined spatial prompts, such as bounding boxes or points provided in the first frame.
2. Using detections from GroundingDINO, where tracking begins automatically once an object matching the text prompt is detected.

The tracking relies on the dynamic memory management approach described in Figure 3.2 to maintain object associations over time. The per-frame processing time T_{frame} can be approximated based on the computational costs of image encoding, mask decoding, memory encoding, and memory attention, according to:

$$T_{\text{frame}} = T_{\text{ie}} + (T_{\text{md}} + T_{\text{me}} + T_{\text{ma}}) \cdot n. \tag{3.1}$$

Where:

T_{ie} : Image encoding time,
T_{md} : Mask decoding time,
T_{me} : Memory encoding time,
T_{ma} : Memory attention time,
n : Number of tracked objects.

The performance of the model depends on the number of objects that are tracked, the image resolution, and the choice of encoders/decoders.

3.2.2 Depth estimation

The internal depth sensing module of the ZED Mini stereo camera provides pixel-wise depth estimation in real-time. This module generates a depth map at the same frequency as the captured stereo image stream, ensuring no added latency relative to image acquisition (see Appendix C.1 for technical specifications). The depth estimation process follows three steps. First, the camera extracts visual features from the left and right images and performs stereo matching. Next, it calculates disparity using a low-resolution cost volume. Finally, it estimates the depth at each pixel using the known stereo geometry and camera intrinsics, as described in Chapter 2.1. The system then upsamples this lower-resolution depth map to match the input image resolution, using upsampling functions that recover high-frequency details.

Depth recovery module

The ZED Mini stereo camera provides real-time depth estimation, but raw depth maps often contain missing or unreliable values that can occur due to the failure of feature matching, textureless regions, or occlusions. These gaps in the depth map appear as holes with undefined values. Since object segmentation is performed using the modified SAM2 architecture, information on the boundaries of the segmented objects is available. A depth recovery module is used to refine depth estimates in segmented object regions based on co-planar hybrid plane fitting [68].

The implemented method assumes that the segmented objects in the scene can be approximated as piecewise-planar surfaces, meaning that pixels belonging to the same object or region often share a common co-planar geometric relationship. The depth recovery module fits local planes to the segmented regions and interpolates missing values. Given a set of valid depth points (x, y, Z) presented within the segmented object region, the objective is to estimate the best-fitting plane that interpolates the missing depth values. A 3D plane is defined as:

$$Z = ax + by + c. \tag{3.2}$$

Where a, b, and c are the plane coefficients that describe the plane's orientation. Different mathematical approaches can be used to solve and find these plane coefficients. The chosen method is selected based on the ratio of missing depth values to the total amount of depth values present in the segmentation mask of a single object (r_{holes}). The choice of plane-fitting method is determined based on this ratio:

$$r_{\text{holes}} = \frac{N_{\text{holes}}}{N_{\text{total}}}. \tag{3.3}$$

If the ratio is within the range $0.7 \leq r_{holes} \leq 0.9$, the plane coefficients are estimated using Random Sample Consensus (RANSAC) [69]. RANSAC is effective in rejecting outliers by iteratively selecting the best set of plane coefficients. However, it is important to note that the RANSAC method is demanding in computational time, and in some cases can converge to a local maximum of a cost function instead of the global maximum. The plane coefficients are found by maximizing the following function:

$$[\hat{a}, \hat{b}, \hat{c}] = \arg\max_{[a,b,c]} \sum_i f_i(a, b, c). \tag{3.4}$$

Where $f_i(a, b, c)$ is defined as:

$$f_i(a, b, c) = \begin{cases} 1, & \text{if } |Z_i - (ax_i + by_i + c)| < T_{\text{residual}} \\ 0, & \text{otherwise} \end{cases}$$

In this research, a residual threshold ($T_{residual}$) of 2 mm is used.

If $r_{\text{holes}} < 0.7$, a Weighted Least Squares Estimation (WSLE) method is used instead. This is a more computationally efficient alternative to RANSAC but is more sensitive to noise. Since the associated confidence levels of the depth values are not available, a less complex weighting approach is used that relies on predefined weights. Since the ZED stereo model already filters out the unreliable depth values, they can be assigned a smaller weight. Valid points are assigned a value of 1, and non-valid values are given a weight of 0.1, ensuring that they contribute minimally to the plane estimation. The plane coefficients are estimated by minimizing the following weighted least-squares function:

$$[\hat{a}, \hat{b}, \hat{c}] = \arg\min_{[a,b,c]} \sum_i w_i \left(Z_i - ax_i - by_i - c\right)^2 \tag{3.5}$$

Where w_i represents the weight assigned to each pixel according to its reliability.

Depth estimates are first normalized before interpolation to stabilize the plane fitting approach [70]. Variations in depth values or the distribution of depth values can lead to poor plane estimations without normalization. The depth map is denormalized after interpolation to match the original absolute scale of the original depth map.

When multiple object masks appear in a frame, performing the refinement (either WLS or RANSAC) sequentially becomes inefficient. Therefore, the framework implements a multi-threaded processing scheme that refines each object mask in parallel. This allows the refinement step to scale better when there are multiple tracked objects.

Chapter 4

Framework evaluation

4.1 Experimental setup

This study evaluates the frameworks ability to produce reliable segmentation and depth outputs by conducting a series of grasping experiments with a physical robotic setup. The experimental setup in the lab consists of a Franka Research 3[1] robotic arm equipped with a ZED Mini stereo camera, mounted using a custom-designed 3D printed camera bracket. Figure 4.1 presents a view of the robotic arm in the experimental setup.

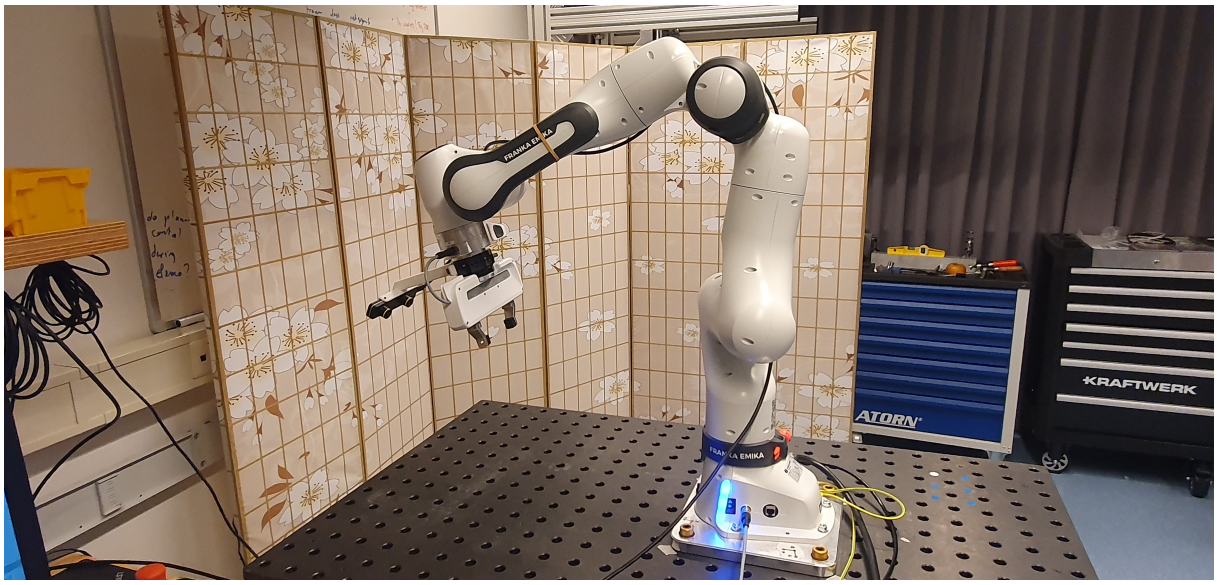


Figure 4.1: *Franka Research 3 robotic arm with a ZED Mini stereo camera mounted via a 3D-printed bracket. This setup was used to evaluate the framework in robotic grasping scenarios.*

Since the evaluation focuses on feasibility through design decisions rather than motion planning, the robot was manually guided to perform a range of grasping actions. The experiments were conducted on a workstation with an NVIDIA GeForce RTX 3070 GPU and an AMD Ryzen 9 CPU.

Detailed specifications of the ZED Mini can be found in Appendix C.1, full hardware specs in Appendix C.2, and a schematic of the camera bracket in Appendix C.3.

[1] `https://franka.de/`

The bracket maintains a fixed camera pose relative to the gripper. The camera can be positioned so that either the left lens, the right lens, or the midpoint between the two aligns with the gripper center. This study used the left lens, as the framework computes disparity and depth relative to it. Data were collected at 720p and 1080p resolutions, with the ZED Mini configured to record at 30 FPS.

4.1.1 Evaluation scenarios

Since the framework uses GroundingDINO for object detection, it supports evaluation on a wide range of objects. However, this study does not aim to test every possible object. Instead, it defines a set of controlled scenarios that reflect challenges relevant to object detection, segmentation, and depth estimation.

1. **Simple Object Grasping**: A scenario with a clearly defined, easily detectable object used to evaluate baseline performance in object detection and grasping.
2. **Translucent Object Detection**: A scenario involving a translucent object used to assess how well the segmentation and depth estimation modules handle transparent materials.
3. **Challenging Background Contrast**: A scenario in which the object blends into the background, designed to test the ability to detect objects with low visual contrast.
4. **Crowded Scene with Multiple Objects**: A scenario containing multiple objects used to evaluate the systems multi-object tracking and segmentation performance in cluttered environments.

Each of these scenarios is evaluated at 720p and 1080p to analyze the impact of resolution on segmentation quality, depth estimation quality, and real-time performance (see Appendix C.1 for resolution specifications).

4.2 Segmentation accuracy evaluation

This evaluation measures segmentation accuracy using the IoU and BIoU metrics defined in Section 2.3. Annotations were created at both 720p and 1080p resolutions. Rather than labeling all the frames present in each evaluation scenario, a subset of 10 representative key frames was selected. These frames were manually chosen to reflect relevant moments, such as occlusions or close object interactions. The key frames should provide a representative measure of overall performance without requiring annotations for every frame in the evaluation scenarios, which is not feasible to obtain in this study.

4.3 Depth accuracy evaluation

ArUco markers can be used to estimate 3D positions and evaluate the accuracy of depth estimation methods in 3D space. The previously described depth accuracy evaluation method described in Section 2.3 is adopted here and is repeated briefly. The depth estimates of the ZED mini and the ArUco-derived depth values are projected into 3D point clouds within the camera coordinate frame. The Euclidean distance between each pair of corresponding 3D points is computed to quantify the error. The Euclidean errors are binned by derived ArUco depth distances and summarized using the RMSE per depth bin. A second-order polynomial and an exponential model are fitted to the resulting RMSE curve to capture the relationship between depth error and distance.

Directly evaluating depth accuracy on curved objects such as apples or water bottles produced unreliable results. Variations in surface geometry caused large differences in absolute depth that

made meaningful comparison impossible. To ensure consistent and interpretable measurements, the evaluation instead used ArUco markers placed on a flat tabletop surface (see Figure 4.2 for a schematic presentation of the setup). Flat surfaces support stable geometry during capture, which improves marker detection and ensures more accurate corner point estimation. This setup enabled precise computation of each markers 3D position in the camera frame, which was then compared to the corresponding 3D coordinates derived from the ZED depth map. Note that this is a separate depth evaluation experiment and is not part of the four scenarios described in Section 4.1.1.

Two separate measurements are performed at both 720p and 1080p resolutions. Measurement 1 uses 6×6 ArUco markers with a size of 60mm. Measurement 2 uses 7×7 ArUco markers with a size of 150 mm. Since the experiments are captured with the Franka Emika Research 3 robotic arm, the maximum allowed depth range is limited to ±855 mm, as specified in the Franka Research 3 datasheet [71]. A depth bin spacing of 20 mm was used in both measurements.

The depth refinement module was not applied because ArUco markers did not produce closed, consistent shapes suitable for segmentation using SAM2. As the module depends on valid object masks, it could not process these scenes. As a result, the refined depth map was excluded from quantitative evaluation. Instead, refinement results are presented qualitatively through visual examples.

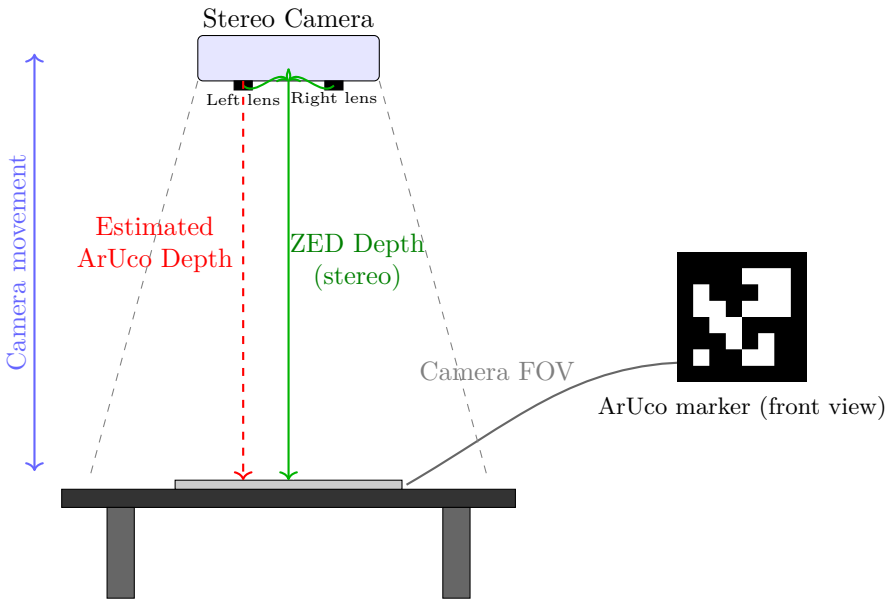


Figure 4.2: *Schematic of the stereo camera setup used for depth accuracy evaluation. The stereo camera looks downward toward an ArUco marker placed on a flat table surface. The camera can move vertically relative to the table. The ZED depth (green arrow) is obtained from the stereo estimation model, while the ArUco depth (red arrow) is derived from pose estimation. Both depth values are later projected into 3D points in the camera coordinate frame (not shown), where their difference is used to compute depth estimation error per bin.*

4.4 Performance evaluation

To assess the real-time capabilities of the framework, the average frame processing time was measured in FPS under various design configurations. Performance is calculated for the three different depth refinement methods integrated in the framework (None, RANSAC, and WLS), as described in Section 3.2.2. Tests were conducted at two input resolutions (720p and 1080p)

and with one, two, and three simultaneously tracked objects. This allows for evaluation of how the depth refinement method, resolution, and tracking affect the performance of the complete framework.

Chapter 5

Results

5.1 Qualitative Evaluation

This section presents the qualitative evaluation of the proposed method across multiple representative test scenarios. The goal is to provide a visual comparison of segmentation and depth estimation under varying conditions.

5.1.1 Overview of Visual Results

An overview of the visual results for the four evaluation scenarios is presented in Figure 5.1. The challenging background contrast scenario includes two test cases: a black controller and a black cup, both placed on the black operating table. The crowded scene scenario includes three separate object compositions, referred to as Fruit Set 1, Fruit Set 2, and Fruit Set 3. A test case label is used to distinguish between different measurements within each scenario. These are indicated in parentheses next to each scenario label along the y-axis in Figure 5.1, and are also listed in the second column of Table 5.1.

5.1.2 Observations per Scenario

Simple object grasping (apple)

This scenario features two brightly colored apples, having high contrast with the background. For both apples, ground truth masks are created. The SAM2 tracking model successfully identifies both apples with a predicted mask that closely aligns with the manually annotated ground truth throughout. The raw depth map from the ZED camera demonstrates clear object-background separation with no missing values due to occlusions or failed feature matching. As a result, the refined depth shows minimal changes. The surface of the objects in the refined depth map is more coplanar because of the depth refinement method.

Translucent object (plastic bottle)

This scenario involves a partially transparent plastic bottle placed on a white table to reduce the contrast between the object and the background. Ground truth masks were created for the bottle. The segmentation model accurately outlines the bottle, although minor deviations are visible near the bottom of the bottle, which was harder to capture because of the shadow. The raw depth map shows many missing values in the outline of the bottle, caused by failed feature matching. This is due to the matching with translucent material and its proximity to the camera. The latter violates the minimum depth range of the ZED, resulting in additional geometric

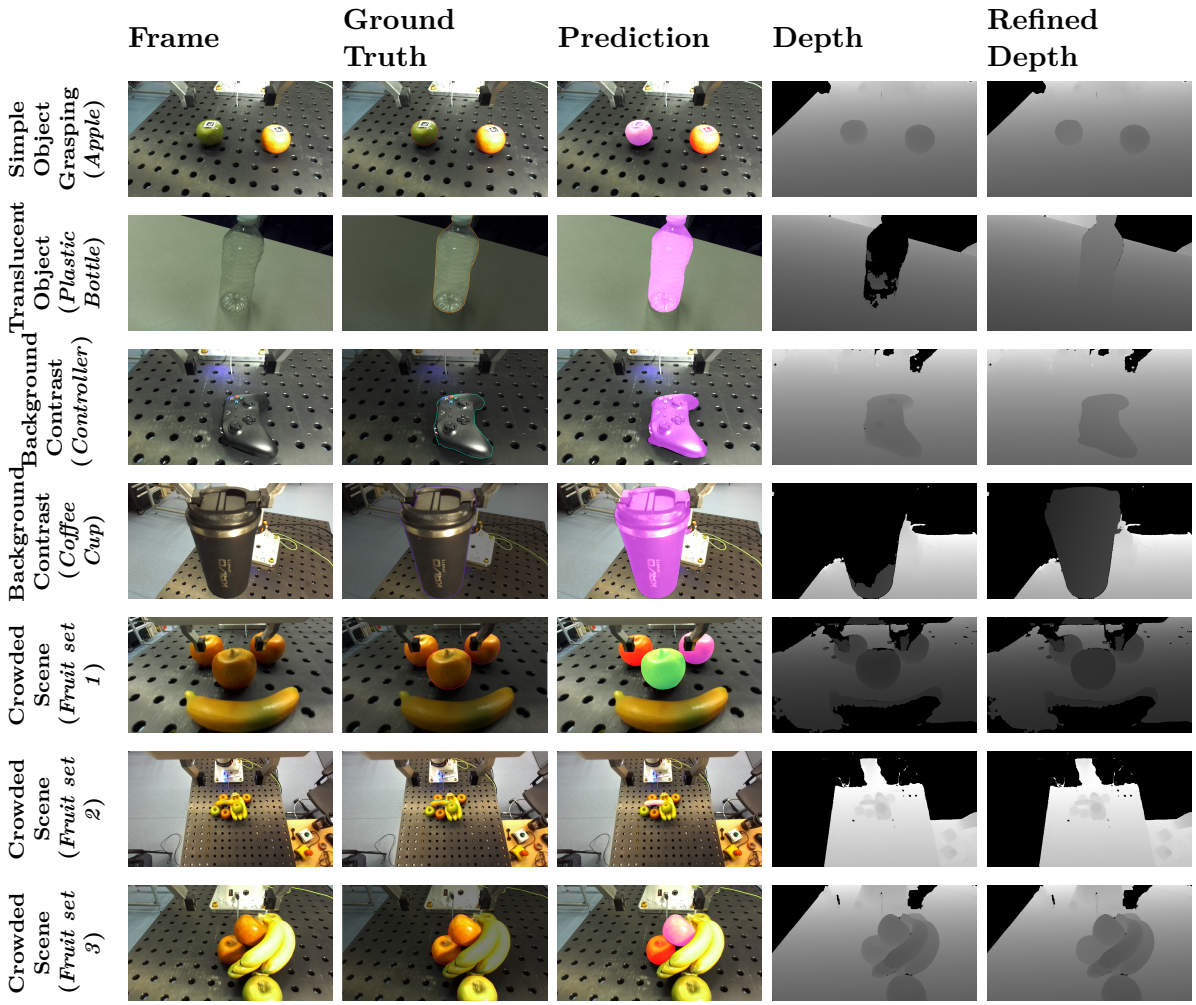


Figure 5.1: *Segmentation, depth estimation, and refinement results across a range of test cases. Each row represents a single test scenario. Columns show, from left to right: the original input frame, the manually created ground truth mask(s), the predicted mask from SAM2, the raw depth map from the ZED camera, and the refined depth map after applying depth post-processing.*

constraints. The refined depth map is capable of filling in the missing values, demonstrating the added value of the refinement step for frames with incomplete raw depth data.

Challenging background contrast (controller)

A black game controller is placed on a black operating table, resulting in minimal contrast between the object and the background. For this test case, the camera must be very close to be able to detect the object using GroundingDINO. This means that for earlier frames where the camera is further away, the object is not detected. Consequently, the tracking is not started, and those frames have missing segmentation masks. In the Figure, the first frame where the controller is detected is shown. From this point onward, the segmentation model can track the object across subsequent frames. The predicted masks are reasonably good estimations, but some parts of the operating table are incorrectly misclassified as part of the controller, as can be seen in the lower-left region in the image. The raw depth map does not suffer from missing values, providing a good estimation. The refined depth map converts the depth values of the object to a co-planar representation, losing finer details around curved surfaces. This can be

seen near the joysticks of the controller.

Challenging background contrast (coffee cup)

Similar to the previous case, this test places a black coffee cup on the black operating table, resulting in minimal contrast. The coffee cup is detected earlier than in the case with the controller, but in some of the initial frames, the coffee cup is not detected. Once the object is recognized, the segmentation masks are generally accurate. However, some misclassifications occur when the object is being grasped by the gripper and when it overlaps with the black parts of the Franka. The black gripper pads are misclassified as part of the cup, and the logo of the Franka is misclassified as well. This can be seen in the figure around the left gripper. The raw depth map contains missing values when the camera is near the object, similar to the previous case. The refined depth map is again capable of estimating some of the missing values by the co-planar depth refinement step.

Crowded scene (fruit set 1)

This scenario features a composition of three apples and one banana placed close, though not in direct contact. Using GroundingDINO, the focus is placed on segmenting only the three apples, while the banana is intentionally excluded. The predicted object masks were highly accurate around the object boundaries of the apples and correctly ignored the banana. The raw depth map was able to capture most of the surfaces of the apples, missing only a few values near the edges. The refined depth map could estimate some of the missing values by the co-planar refinement step.

Crowded scene (fruit set 2)

The second crowded fruit scene includes a more densely packed composition of fruits. Four apples, one single banana, and a banana bunch are placed together. In this case, GroundingDINO is prompted specifically with the text prompt single banana to evaluate its ability to distinguish the target object from similar objects. The predicted mask accurately captures the single banana, even in frames where it overlaps with the banana bunch. The raw depth map appears to capture the scene reasonably well. Since in this test case the gripper never moves close to the object in the scene but hovers above it, no difference can be seen visually between the raw and refined depth maps.

Crowded scene (fruit set 3)

This third crowded fruit scene includes a composition of three overlapping apples with the banana bunch. The goal in this case is to track the three apples. In the initial frames, only the two red apples are visible, the green apple is not yet in the frame. As a result, tracking is initiated based on the detection of the two red apples, and the green apple is never included since the system does not add new objects to the tracking once it has started. The raw depth map presents an accurate representation of the depth surfaces of the objects. By performing refinement, the estimated depth surfaces are more co-planar than the original raw depth map.

5.2 Quantitative Evaluation

This section presents quantitative results to complement the qualitative results. The segmentation accuracy, depth estimation accuracy, and runtime performance are reported to asses the proposed framework.

5.2.1 Segmentation Metrics (IoU, BIoU)

Segmentation accuracy is evaluated for each of the test cases shown in Figure 5.1. Each test case includes a representative keyframe showing both the predicted and manually annotated ground truth masks of the tracked object(s). For the quantitative evaluation, 10 keyframes per test case were manually annotated at both 720p and 1080p resolutions. The IoU and BIoU scores reported in Table 5.1 are the average values across these annotated frames for each test case.

Table 5.1: *Segmentation performance metrics (IoU and BIoU) at 720p and 1080p resolution for each test case. Results are averaged over 10 manually annotated keyframes per test case.*

Scenario	Test Case	IoU 720p	IoU 1080p	BIoU 720p	BIoU 1080p
Simple Object Grasping	Apple	0.91	0.95	0.88	0.91
Translucent Object Detection	Plastic Bottle	0.90	0.92	0.85	0.88
Challenging Background Contrast	Controller	0.55	0.60	0.45	0.48
	Coffee Cup	0.78	0.81	0.64	0.69
Crowded Scene with Multiple Objects	Fruit Set 1	0.90	0.94	0.86	0.88
	Fruit Set 2	0.85	0.88	0.76	0.78
	Fruit set 3	0.60	0.64	0.54	0.56

The calculated segmentation metrics complement the qualitative observations discussed in the previous section. Overall, the results show high segmentation accuracy in scenarios with clear contrast between objects and the background, and in less visually complex compositions. For example, the test case with clear contrast (Simple Object Grasping) achieves the highest metric scores across both resolutions. This aligns well with visual observations, where the segmentation predictions for both apples are consistently accurate. Similarly, the translucent object case with the plastic bottle shows high accuracy for both resolutions, despite the challenges introduced by translucency.

In contrast, accuracy drops substantially in the Controller case, where low background contrast reduces detection reliability. The IoU of 0.55 at 720p and 0.66 at 1080p confirms the visual observation that object detection fails in early frames. Additionally, the predicted mask includes misclassified background regions, which further reduces the BIoU due to poor boundary precision. The Coffee Cup case performs better due to earlier detection and fewer misclassifications between the cup and the table. However, misclassification of gripper pads and the Franka still lowers the boundary accuracy, which is reflected in the reduced values of BIoU.

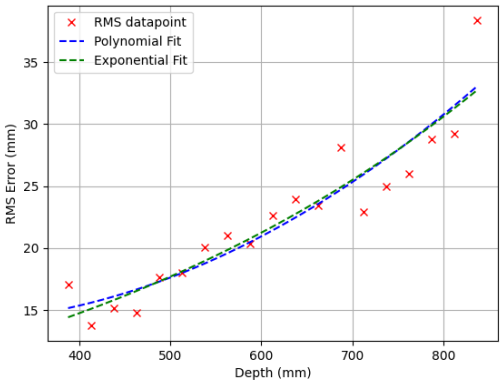
In crowded scenes, the Fruit Set 1 test case maintains high segmentation quality, indicating that the model effectively handles cases where objects are in close proximity but are not touching. Performance slightly drops in Fruit Set 2, where a denser composition and object overlap introduce more challenging segmentation conditions. The lower BIoU values suggest that the primary source of error is the accuracy of the boundaries. The lowest scores are observed in Fruit Set 3. In this test case, the third apple is never included in the tracking due to being outside the view when the tracking is started. This limitation is reflected in the lower scores, even when the two other apples are segmented correctly.

Overall, the quantitative results complement the qualitative findings: segmentation accuracy is highest when contrast is strong and object compositions are uncluttered, and it decreases in scenes with poor visibility, overlapping objects, or missed detections.

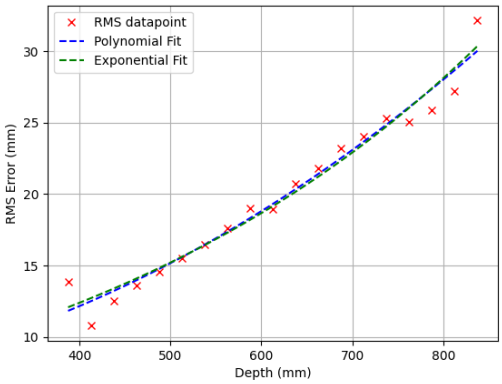
5.2.2 Depth Accuracy Evaluation

Figure 5.2 presents the depth estimation error for the separate experimental setup using two different ArUco markers, as described in Section 4.3. The results show RMSE Euclidean distance of depth estimates as a function of the derived ArUco depth distance. Each subplot compares the values of RMSE (red scatter) with a polynomial fit (blue line) and an exponential fit (green dashed line) to model the trend of the error. As expected, the depth error increases non-linearly with distance from the camera. Both fitted models capture this trend well across the tested depth range.

Experiment 1: 6×6 ArUco markers (60 mm)

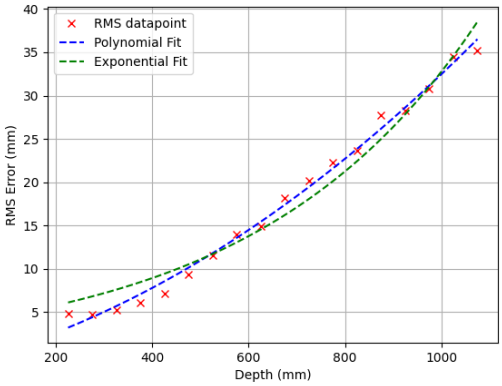


(a) *720p resolution*

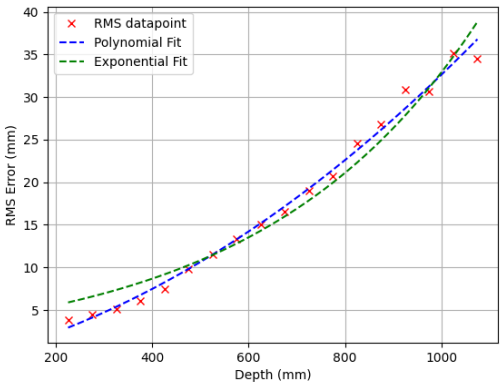


(b) *1080p resolution*

Experiment 2: 7×7 ArUco markers (150 mm)



(c) *720p resolution*



(d) *1080p resolution*

Figure 5.2: *Comparison of depth error behavior across two experimental setups using different ArUco marker types and sizes. Experiment 1 uses 6×6 ArUco markers with a size of 60 mm, while Experiment 2 uses 7×7 ArUco markers with a size of 150 mm. Each setup is evaluated for 720p and 1080p. RMSE data points are shown as red scatter points per depth bin. Data points are fitted using a polynomial (blue) and exponential (green dashed) model, estimating the error trend.*

In Experiment 1 (6×6 markers, 60 mm), the error increases more steeply with distance. The difference between 720p and 1080p resolutions is small, but the 1080p resolution shows slightly lower errors and variance. The polynomial and exponential fits closely follow the RMSE trend, with minimal difference between the two.

In Experiment 2 (7×7 markers, 150 mm), the overall error is lower throughout the depth range compared to the results of Experiment 1. The larger marker size improves detection stability, contributing to more accurate depth estimation. As in experiment 1, the 1080p resolution also shows lower errors and variance. Both the polynomial and exponential models again approximate the error distribution well, with small deviations at the lowest and highest depth values.

The results confirm that increasing both image resolution and marker size improves depth accuracy. The fitted models reliably capture the observed error trend and provide insight into how accuracy degrades with distance.

5.2.3 Performance Evaluation

The real-time performance of the framework was evaluated by measuring the FPS under different conditions. In Table 5.2, the results are shown for scenarios with 1, 2, or 3 tracked objects, in both 720p and 1080p resolutions. Three refinement configurations were tested: no depth refinement, RANSAC-based refinement, and WLS-based refinement.

Table 5.2: *Performance of the full framework measured in FPS, using no refinement, RANSAC refinement, and WLS refinement. Results are captured for 720p and 1080p resolutions, with 1, 2, or 3 tracked objects.*

Resolution	Refinement Method	1 Object (FPS)	2 Objects (FPS)	3 Objects (FPS)
720p	None	24-27	19-21	14-16
	RANSAC	6-8	5-7	3-5
	WLS	10-12	8-11	6-7
1080p	None	21-23	16-21	12-14
	RANSAC	5-6	3-6	2-4
	WLS	8-11	7-10	5-7

As expected, the highest performance is achieved when no refinement is applied. This is because no extra steps are applied to the original depth map extracted by the ZED Mini. As discussed in Section 3.2.2, the raw depth map is retrieved at the same frequency as the stereo image stream, introducing no additional latency. As a result, it is possible to maintain high frame rates, even when tracking multiple objects.

When applying depth refinement, the additional computational cost increases. The RANSAC-based refinement method introduces the largest drop in performance. This result is as expected, as the RANSAC-based plane fitting performs iterative optimization to achieve the best plane estimate. Performing this for multiple objects in the scene increases computational costs, even when processing using multi-threading.

The performance of the WLS-based refinement is twice as fast as that of the RANSAC method. However, the frame rates remain less than half of those achieved without any refinement, indicating the trade-off between depth quality and performance speed.

A decrease in FPS is observed as the number of tracked objects increases. This is partially due to the increased computational costs for the per-frame processing time introduced by the

SAM2 memory module (see equation 3.1). The other factor influencing performance is the image resolution: 720p consistently outperforms 1080p. The lower resolution reduces the number of pixels per mask, reducing both processing time and refinement time per frame.

Chapter 6

Discussion

6.1 General Discussion

This study evaluated the feasibility of integrating segmentation, object tracking, and stereo-based depth estimation into a modular real-time vision framework for object-level robotic interaction. The framework successfully processed live camera input while generalizing to previously unseen environments and objects. Prompt-based segmentation using SAM2 enabled zero-shot performance, while the ZED stereo-vision depth estimation model provided absolute depth estimation without requiring scene-specific retraining. The integrated co-planar depth refinement module is implemented as an optional post-processing step to improve object-level depth estimations. The modular design of the framework supports flexible integration by allowing depth estimation, segmentation, and refinement components to be substituted independently with minimal adjustments.

The first subgoal assessed whether the framework could produce reliable segmentation and depth output for a downstream robotic grasping task without requiring environment-specific tuning. A set of robotic grasping test scenarios was used to evaluate the framework. These scenarios allowed feasibility assessment of the framework for these robotic interaction tasks, without relying on task-specific training.

The framework achieved consistently high segmentation accuracy in clean, high-contrast scenes, achieving average IoU scores of up to 0.95 and BIoU of up to 0.91, indicating good boundary precision. It also performed well in the Translucent Object test case, with average IoU and BIoU scores of up to 0.92 and 0.88, respectively, despite the challenges introduced by the translucency of the object. In more challenging settings, such as cluttered backgrounds or crowded scenes, segmentation performance declined. For example, in the Controller test case with poor background contrast, the average IoU dropped to 0.60 and the average BIoU to 0.48. Similar declines were seen in the Fruit Set 3 test case, where tracking failed for one object because it was not detected at the moment of tracking initialization. The system maintained reasonable accuracy in moderately complex environments with good contrast.

The translucent object scenario revealed missing depth values caused by weak stereo feature matching. The depth refinement module recovered some of these missing values, but the co-planar assumption oversimplified curved surfaces. Refinement also introduced additional computational overhead.

A separate experiment was conducted using ArUco markers placed on a flat tabletop to quantify the ZED depth estimation. Two marker sizes were tested at both 720p and 1080p resolution. Results showed that depth error increased non-linearly with distance. 1080p resolution consistently reduced both error and variance. Additionally, using larger markers improved detection accu-

racy by increasing the number of observable pixels. These findings confirm that the raw ZED depth output is reliable within the working range of the robot and does not require additional tuning.

Although the framework demonstrated generalization across the tested scenarios, all experiments were conducted in a controlled lab environment. Its ability to generalize to more different settings and conditions remains unverified in this study. The evaluation did not include a quantitative accuracy assessment of the refined depth map, since the refinement module depends on valid segmentation masks that could not be obtained for ArUco markers because they did not have closed, consistent shapes. As a result, depth accuracy measurements focused solely on the ZED depth output. Depth refinement was therefore only evaluated qualitatively.

The second subgoal focused on identifying design strategies for segmentation-based tracking on a live camera stream while preserving real-time performance. SAM2 was selected for its integrated tracking capabilities, avoiding the need for external tracking models such as XMEM or Cutie, which increase latency. To support frame-by-frame prediction on live camera streams, the memory module was adapted. The modified version removed the dependency on future frames and updated memory only when valid segmentation results were available. This selective update strategy prevented empty states from overwriting reliable past states. As a result, the tracker successfully re-identified objects after occlusion, provided they retained similar. The design changes made to SAM2 enabled real-time segmentation-based tracking using live camera streams.

The third subgoal explored how image resolution and the number of tracked objects affect the real-time performance of the framework. Increasing the number of tracked objects reduced frame rates due to the added computational cost of memory operations and mask predictions. This study stated a lower threshold of 10 FPS as sufficient for the real-time robotic grasping task. At 720p resolution, the system maintained real-time performance above 10 FPS, reaching 24 to 27 FPS with one tracked object and 14 to 16 FPS with three objects. At 1080p, the system achieved 21 to 23 FPS for one object and 12 to 14 FPS for three objects. By enabling the optional depth refinement module, performance declined. RANSAC-based refinement introduced the highest latency, dropping frame rates as low as 3 to 5 FPS at 720p and 2 to 4 FPS at 1080p when tracking three objects. WLS-based refinement was faster, reaching 10-12 FPS at 720p and 8 to 11 FPS at 1080p with one object. However, WLS also fell below the 10 FPS threshold when processing more than two tracked objects.

6.2 Limitations

Several limitations of the designed framework are important to highlight. First, as the number of tracked objects increases, the processing time increases due to the extra memory and prediction operations required by SAM2. Latency increases even more when depth refinement is enabled, especially with the RANSAC-based method. Although multithreading helps mitigate latency to some extent, real-time performance quickly degrades as more objects are added.

Second, the framework relies on accurate prompt-based initialization. If an object is not correctly segmented or detected at the start, it will not be included in the memory bank and cannot be tracked or recovered later. This limitation can be critical in cluttered scenes where objects can be occluded during the initialization or when objects come into the frame after tracking has already started. Since initialization often depends on GroundingDINO detections, any missed detections directly prevent the tracking of those missed objects. This limitation is somewhat mitigated as the framework allows tracker re-initialization during runtime. The user is allowed to reset the memory and restart prompt initialization without restarting the camera stream or rebuilding models.

Third, the depth refinement module assumes a co-planar structure for segmented objects. This assumption works well for flat surfaces, but does not generalize well to curved shapes. In such cases, the estimated planes can lead to oversimplified or inaccurate depth estimations. It is recommended to use refinement selectively. Enable it only for frames where missing depth values occur in important objects or frames. In addition, refinement can be toggled on and off during runtime, allowing selective usage during streaming.

Finally, all experiments took place in a controlled indoor lab setting. Although the evaluation included a range of test objects and the selected models support zero-shot estimation, this study did not confirm whether the framework generalizes to more different environments. Further validation in more diverse and dynamic settings is needed to support stronger claims about its generalization capability.

Chapter 7

Conclusion

This study developed a modular, real-time vision framework that integrates segmentation, object tracking, and stereo-based depth estimation for object-level robotic interaction. The framework adapts the prompt-driven SAM2 segmentation model to operate on live camera streams without access to future frames, enabling real-time tracking. The ZED Mini stereo camera provides dense depth estimates with zero-shot generalization, and an optional depth refinement module enhances depth completeness in segmented regions through co-planar hybrid plane fitting.

Experimental evaluation across a range of robotic grasping scenarios demonstrated that the framework delivers reliable object segmentation and depth estimation without environment-specific tuning. Segmentation accuracy remained high in clean, high-contrast scenes and moderately complex environments. The depth module produced stable estimates across the working range of the robot, with higher accuracy at increased resolution. However, performance declined in cluttered scenes, and the refinement step introduced substantial computational overhead, particularly when tracking multiple objects.

The framework maintained real-time performance under constrained configurations, achieving frame rates above the 10 FPS threshold with one to three tracked objects at 720p and 1080p. Its modular structure allows components to be replaced or extended independently, making it adaptable to new use cases and hardware setups. Although the framework generalized well across the different objects and settings in the test scenarios, its generalization capability remains unverified in more diverse settings.

7.1 Recommendations and future work

Future work can expand in several directions:

- **Conversion to object-level point clouds:** The next step combines the dense depth maps with the predicted segmentation masks to generate point clouds for individual objects. This enables 3D visual feedback derived from 2D promptable segmentation using SAM2 as a foundation model. The resulting object-level point clouds provide structured input for robotic systems to perceive and interact with objects in dynamic scenes.
- **Improving depth refinement:** The refinement module currently poses the largest computational bottleneck. Replacing plane-fitting with more efficient post-processing methods could reduce the computational overhead introduced by the methods.
- **Component replacement:** The modular design of the framework enables easy substitution of components. For example, GroundingDINO can be replaced with a more accurate vision-language model that produces bounding boxes to initialize SAM2 tracking. Stronger

detectors may improve performance in low-contrast or crowded scenes. Since detection is only performed during tracking initialization, using a more precise model enhances tracking quality without significantly increasing the computational cost during runtime

- **Generalization testing:** Further validation in unstructured or real-world robotic settings is needed to confirm the generalization performance of the framework under more varying conditions.

Bibliography

[1] Andrew J. Davison. Futuremapping: The computational structure of spatial ai systems. *arXiv preprint arXiv:1803.11288*, 3 2018.

[2] Nitin Sharma, Jitendra Kumar Pandey, and Surajit Mondal. A review of mobile robots: Applications and future prospect. *International Journal of Precision Engineering and Manufacturing*, 24:1695–1706, 9 2023.

[3] Md Tanzil Shahria, Md Samiul Haque Sunny, Md Ishrak Islam Zarif, Jawhar Ghommam, Sheikh Iqbal Ahamed, and Mohammad H. Rahman. A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions. *Robotics 2022, Vol. 11, Page 139*, 11:139, 12 2022.

[4] Nicole Robinson, Brendan Tidd, Dylan Campbell, Dana Kuli, and Peter Corke. Robotic vision for human-robot interaction and collaboration: A survey and systematic review. *ACM Transactions on Human-Robot Interaction*, 12, 2 2023.

[5] Kihong Park, Seungryong Kim, and Kwanghoon Sohn. High-precision depth estimation with the 3d lidar and stereo fusion. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2156–2163, 9 2018.

[6] Minseung Lee, Seokha Moon, Seung Joon Lee, and Jinkyu Kim. Sparse-to-dense lidar point generation by lidar-camera fusion for 3d object detection. *arXiv preprint arXiv:2409.14985*, 9 2024.

[7] M A U ; Khan, D ; Nazir, A ; Pagani, H ; Mokayed, M ; Liwicki, D ; Stricker, M Z A Afzal, Carlo Alberto Avizzano, Muhammad Ahmed, Ullah Khan, Danish Nazir, Alain Pagani, Hamam Mokayed, Marcus Liwicki, Didier Stricker, and Muhammad Zeshan Afzal. A comprehensive survey of depth completion approaches. *Sensors 2022, Vol. 22, Page 6969*, 22:6969, 9 2022.

[8] Clemens Linnhoff, Kristof Hofrichter, Lukas Elster, Philipp Rosenberger, and Hermann Winner. Measuring the influence of environmental conditions on automotive lidar sensors. *Sensors (Basel, Switzerland)*, 22:5266, 7 2022.

[9] Matthew Levine. Towards safe, real-time systems: Stereo vs images and lidar for 3d object detection. *arXiv preprint arXiv:2202.12773*, 2 2022.

[10] Antoine Mauri, Redouane Khemmar, Benoit Decoux, Tahar Benmoumen, Madjid Haddad, and Rémi Boutteau. A comparative study of deep learning-based depth estimation approaches: Application to smart mobility. pages 80–84, 9 2021.

[11] Jure bontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1592–1599, 6 2015.

[12] Hamid Laga, Laurent Valentin Jospin, Farid Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(4):1738–1764, 4 2022.

[13] Stefan Ainetter and Friedrich Fraundorfer. End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb. *Proceedings - IEEE International Conference on Robotics and Automation*, 2021-May:13452–13458, 2021.

[14] Cebin Fu, Xiangyan Tang, Yue Yang, Chengchun Ruan, and Binbin Li. A survey of research progresses on instance segmentation based on deep learning. *Communications in Computer and Information Science*, 2099 CCIS:138–151, 2024.

[15] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 9 2020.

[16] Maria Tzelepi and Anastasios Tefas. Semantic scene segmentation for robotics applications. In *2021 12th International Conference on Information, Intelligence, Systems Applications (IISA)*, pages 1–4, 8 2021.

[17] Stefan Ainetter, Christoph Böhm, Rohit Dhakate, Stephan Weiss, and Friedrich Fraundorfer. Depth-aware object segmentation and grasp detection for robotic picking tasks. *32nd British Machine Vision Conference, BMVC 2021*, 11 2021.

[18] Wenqi Ren, Yang Tang, Qiyu Sun, Chaoqiang Zhao, and Qing Long Han. Visual semantic segmentation based on few/zero-shot learning: An overview. *IEEE/CAA Journal of Automatica Sinica*, 11:1106–1126, 11 2022.

[19] Xiangtai Li, Henghui Ding, Haobo Yuan, Wenwei Zhang, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-based visual segmentation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4 2023.

[20] Xing Zi, Kairui Jin, Xian Tao, Jun Li, Ali Braytee, Rajiv Ratn Shah, and Mukesh Prasad. Visual and text prompt segmentation: A novel multi-model framework for remote sensing. *arXiv preprint arXiv:2503.07911*, 3 2025.

[21] Rui Yao, Guosheng Lin, Shixiong Xia, Jiaqi Zhao, and Yong Zhou. Video object segmentation and tracking: A survey. *arXiv preprint arXiv:1904.09172*, 4 2019.

[22] Arnav Acharyya, Dustin Hudson, Ka Wai Chen, Tianjia Feng, Chih Yin Kan, and Truong Nguyen. Depth estimation from focus and disparity. *Proceedings - International Conference on Image Processing, ICIP*, 2016-August:3444–3448, 8 2016.

[23] Eun-Kyung Lee, Seung-Uk Yoon, and Yo-Sung Ho. Generation of multiple depth images from a single depth map using multi-baseline information. 2008.

[24] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.

[25] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001.

[26] Woo-Seok Jang and Yo-Sung Ho. Efficient disparity map estimation using occlusion handling for various 3d multimedia applications. *IEEE Transactions on Consumer Electronics*, 57(4):1937–1943, 2011.

[27] Yingqian Wang, Longguang Wang, Jungang Yang, Wei An, and Yulan Guo. Flickr1024: A large-scale dataset for stereo image super-resolution. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3852–3857, 2019.

[28] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2811–2820, 12 2017.

[29] Qingsong Yan, Qiang Wang, Kaiyong Zhao, Bo Li, Xiaowen Chu, and Fei Deng. Rethinking disparity: A depth range free multi-view stereo based on disparity. *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, 37:3091–3099, 11 2022.

[30] Junming Zhang, Katherine A. Skinner, Ram Vasudevan, and Matthew Johnson-Roberson. Dispsegnet: Leveraging semantics for end-to-end learning of disparity estimation from stereo imagery. *IEEE Robotics and Automation Letters*, 4:1162–1169, 9 2018.

[31] Zhenyao Wu, Xinyi Wu, Xiaoping Zhang, Song Wang, and Lili Ju. Semantic stereo matching with pyramid cost volumes. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:7483–7492, 10 2019.

[32] Xiaowei Yang, Yong Zhao, Zhiguo Feng, Haiwei Sang, Zhenbo Zhang, Guiying Zhang, and Lin He. A light-weight stereo matching network based on multi-scale features fusion and robust disparity refinement. *IET Image Processing*, 17:1797–1811, 5 2023.

[33] Jinlong Yang, Cheng Wu, Gang Wang, and Dong Chen. Guided aggregation and disparity refinement for real-time stereo matching. *Signal, Image and Video Processing*, 18:4467–4477, 7 2024.

[34] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:3523–3542, 1 2020.

[35] Gabriela Csurka, Riccardo Volpi, and Boris Chidlovskii. Semantic image segmentation: Two decades of research. *Foundations and Trends in Computer Graphics and Vision*, 14:1–162, 2 2022.

[36] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:640–651, 11 2014.

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241, 11 2015.

[38] Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, and Damayanthi Herath. Semantic segmentation using vision transformers: A survey. *Engineering Applications of Artificial Intelligence*, 126, 5 2023.

[39] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6877–6886, 12 2020.

[40] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022-June:1280–1289, 12 2021.

[41] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE International Conference on Computer Vision*, pages 9630–9640, 4 2021.

[42] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:386–397, 3 2017.

[43] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9156–9165, 2019.

[44] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12363 LNCS:649–665, 12 2019.

[45] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12346 LNCS:213–229, 5 2020.

[46] Jie Hu, Liujuan Cao, Yao Lu, ShengChuan Zhang, Yan Wang, Ke Li, Feiyue Huang, Ling Shao, and Rongrong Ji. Istr: End-to-end instance segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8660–8669. IEEE, 2021.

[47] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE International Conference on Computer Vision*, pages 9992–10002, 3 2021.

[48] Anurag Arnab, Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Måns Larsson, Alexander Kirillov, Bogdan Savchynskyy, Carsten Rother, Fredrik Kahl, and Philip H.S. Torr. Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction. *IEEE Signal Processing Magazine*, 35:37–52, 1 2018.

[49] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *Proceedings of the IEEE International Conference on Computer Vision*, pages 3992–4003, 4 2023.

[50] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, Christoph Feichtenhofer, and Meta Fair. Sam 2: Segment anything in images and videos. 8 2024.

[51] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *Proceedings of Machine Learning Research*, 139:8748–8763, 2 2021.

[52] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2022.

[53] Zhuoyang Zhang, Han Cai, and Song Han. Efficientvit-sam: Accelerated segment anything model without performance loss. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7859–7863, 2024.

[54] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 6 2023.

[55] Han Shu, Wenshuo Li, Yehui Tang, Yiman Zhang, Yihao Chen, Houqiang Li, Yunhe Wang, and Xinghao Chen. Tinysam: Pushing the envelope for efficient segment anything model. *arXiv preprint arXiv:2312.13789*, 12 2023.

[56] Chong Zhou, Xiangtai Li, Chen Change Loy, and Bo Dai. Edgesam: Prompt-in-the-loop distillation for on-device deployment of sam. *arXiv preprint arXiv:2312.06660*, 12 2023.

[57] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 6 2023.

[58] Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13688 LNCS:640–658, 7 2022.

[59] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. *arXiv preprint arXiv:2310.12982*, 10 2023.

[60] Chaitanya Ryali, Yuan Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, Jitendra Malik, Yanghao Li, and Christoph Feichtenhofer. Hiera: A hierarchical vision transformer without the bells-and-whistles. *Proceedings of Machine Learning Research*, 202:29441–29454, 6 2023.

[61] Jialun Pei, Zhangjun Zhou, and Tiantian Zhang. Evaluation study on sam 2 for class-agnostic instance-level segmentation. *arXiv preprint arXiv:2409.02567*, 9 2024.

[62] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47:2280–2292, 6 2014.

[63] Qian She and Hongyang Yu. A general means for depth data error estimation of depth sensors. *Advances in Computer Science Research*, pages 418–425, 5 2019.

[64] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 6 1981.

[65] Bowen Cheng, Ross Girshick, Piotr Dollár, Alexander C. Berg, and Alexander Kirillov. Boundary iou: Improving object-centric image segmentation evaluation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 15329–15337, 3 2021.

[66] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 3 2023.

[67] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, 10 2018.

[68] Lingfeng Xu, Oscar C. Au, Wenxiu Sun, Yujun Li, and Jiali Li. Hybrid plane fitting for depth estimation. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4, 2012.

[69] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381395, 6 1981.

[70] Nils Einecke and Julian Eggert. Evaluation of direct plane fitting for depth and parameter estimation. In *2010 International Conference on Digital Image Computing: Techniques and Applications*, pages 492–497, 2010.

[71] Franka Emika. *Franka Research 3 Datasheet*, August 2022. Version 1.1, Document No. 120020. Available at: `https://download.franka.de/documents/120020_Datasheet%20Franka%20Research%203_1.1_EN.pdf`.

[72] Stereolabs. *ZED Mini Camera and SDK Overview*, 2024. Available: `https://www.generationrobots.com/media/zed-mini-camera-datasheet.pdf`.

A Statement about usage of AI tools in scientific writing

Statement about the usage of AI tools for writing as required by the official 'Use of AI in Education at the University of Twente[1]' page.

During the preparation of this work, the author used the following tools for the following reasons:

- *Writefull:* Used to refine the language, check grammar, and improve academic writing. The used model is the official Writefull Language model for scientific writing.
- *ChatGPT/Deepseek:* In some cases, ChatGPT or Deepseek is used to check and improve scientific writing. It is also used to assist in structuring sections to improve the overall flow of this thesis.

After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

[1] Utwente information available at: `https://www.utwente.nl/en/learning-teaching/Expertise/ai-in-education/use-of-ai-in-education-at-the-university-of-twente.pdf`

B Extra detailed information on SAM2

B.1 SAM2 detailed overview

A detailed overview of the SAM2 workflow is presented in Figure B. This diagram is constructed based on the SAM2 source code and the information presented in the official paper [50]. The components of the model are discussed in this section.

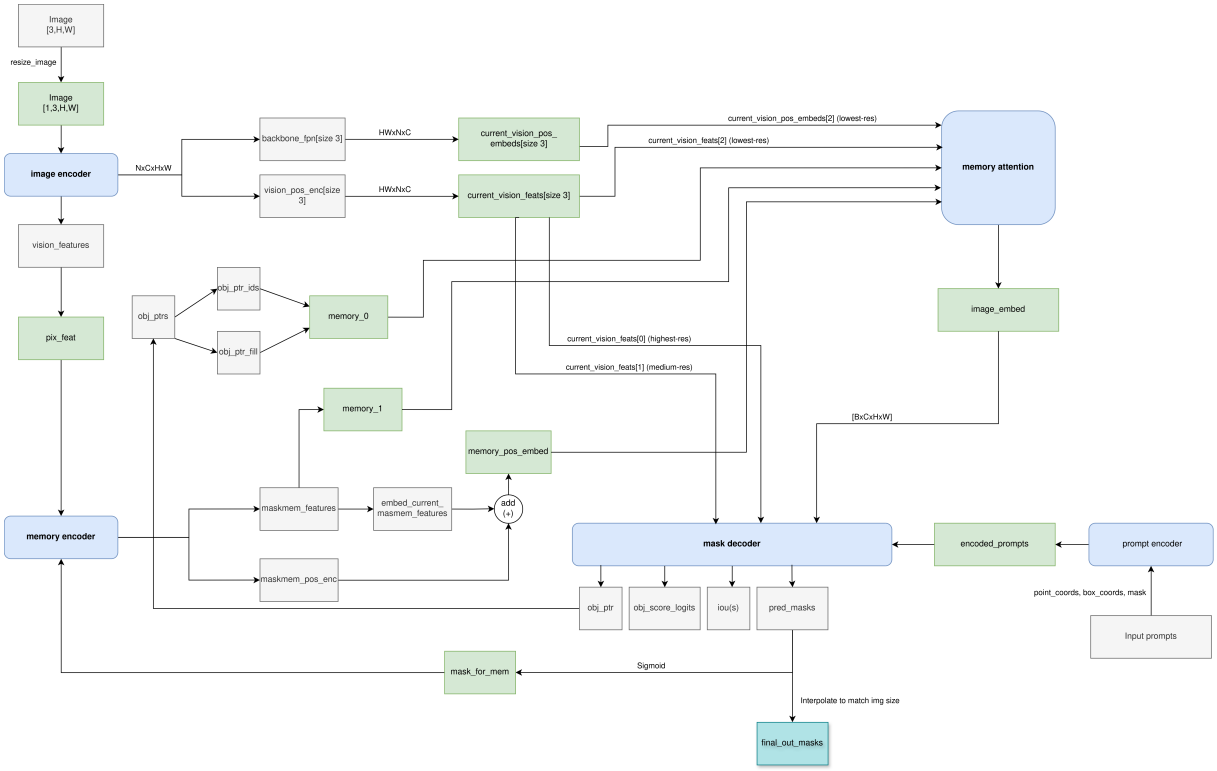


Figure B.1: *Full diagram detailed workflow of SAM2.*

B.1.1 Input Image Processing

The workflow begins with an input image of shape $[3, H, W]$, which is resized to match the input size of the image encoder. The image encoder extracts vision features, producing an output tensor of shape $N \times C \times H \times W$. These features are then processed further by a feature pyramid network backbone and positional encoding module, generating:

- **Current vision features**: Vision features at multiple resolutions (high, medium, and low). The medium and high-resolution vision features are used in the mask decoder.
- **Current vision positional embeddings**: The vision positional embeddings used retain spatial information at multiple resolutions (high, medium, and low). They essentially help the model understand where the detected features are located relative to the entire

image. Only low-resolution position embeddings are used for memory attention to keep the process efficient.

B.1.2 Memory Encoding and Attention Mechanism

The memory mechanism enables the ability to take advantage of past information to refine current segmentation. The extracted vision features from the image encoder are transformed into pixel features before being processed by the memory encoder. This allows the same features from the image encoder to be reused, eliminating the need for additional memory models. The pixel features are combined with memory positional encodings before being used in memory attention, where they interact with previously stored memory representations through cross-attention to improve segmentation consistency over time.

Object pointers are used to associate existing and newly detected masks. This association is based on the calculated IoU score and the object score logits, which account for similarity and occlusions in a quantifiable metric for association. If the number of available object pointers is less than the maximum allowed memory size, empty pointers (obj_ptr_fill) are added to maintain a constant input size required by the memory attention.

B.1.3 Prompt Encoding

Allowed input prompts can be point coordinates, bounding box coordinates, or mask inputs. Note that mask outputs are used as new inputs in subsequent frames as the output of SAM2 is segmentation masks. Prompts are encoded by and passed to the mask decoder, where they interact with image embeddings and/or memory features.

B.1.4 Mask Decoding and Prediction

The mask decoder combines information from multiple sources (vision features, memory features, and encoded prompts) to predict segmentation masks. The outputs of the mask decoder are:

- **Object pointer scores**, used in object association
- **IoU**, estimating the precision of the detected masks.
- **Predicted masks**, based on multi-scale features and memory.

The predicted masks can be of different sizes than needed relative to the original input image size. This depends on the chosen model architecture parameters that influence the final output size. As a final step, the predicted masks are interpolated to match the input image size.

C Experimental Setup and Specifications

C.1 ZED-Mini specifications

Table C.1: *ZED Mini Camera Specifications [72].*

Technical Specifications	**Values**
Output Resolution	2x (2208Œ1242) @15fps 2x (1920x1080) @30fps 2x (1280x720) @60fps 2x (672x376) @100fps
Field of View	Max. 102(H) x 57(V) x 118(D)
RGB Sensor	1/3 4MP CMOS
Focal Length	3.06 mm
Depth Sensing Information	
Baseline Distance	63 mm
Depth Range	0.15m to 15m
Depth Map Resolution	Equal to image resolution

C.2 Hardware specifications

The hardware used in the experimental setup includes the following components:

Table C.2: *Hardware Specifications*

Component	**Specification**
GPU	NVIDIA GeForce RTX 3070
CUDA Cores	5888
VRAM	8 GB GDDR6
CPU	AMD Ryzen 9 6950x 16-core
RAM	24
Operating System	Ubuntu 22.04 LTS
Stereo Camera	ZED Mini (See Table C.1 for full specs)
Robotic Arm	Franka Emika Research 3

C.3 3D printed camera bracket

Figure C.1 contains a schematic side view with distances of the 3D printed camera bracket used for the ZED Mini. It can be seen that the distance between the gripper and camera mount is greater than the minimum distance of 15 mm (Table C.1). Special thanks to Gert Jan for designing the camera bracket.

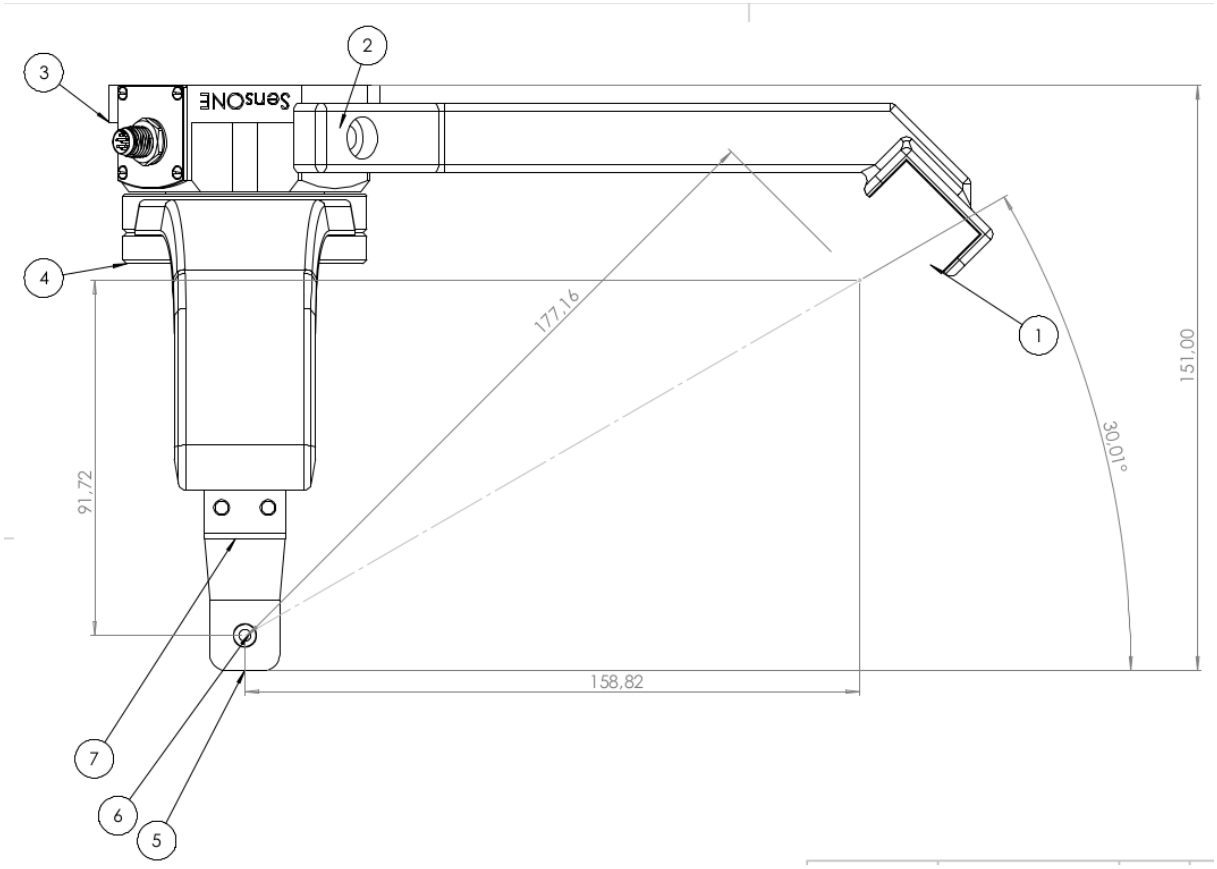


Figure C.1: *Schematic side view of the 3D-printed camera bracket mounted on the gripper. The design ensures that the camera remains above the minimum depth range of 150 mm.*