UNIVERSITAT DE BARCELONA

Bachelor's Thesis

Degree in Computer Science

Faculty of Mathematics and Computer Science

Universitat de Barcelona

FoodMem: A Fast and Precise Food Video Segmentation

Author: Adrián Galán Pacheco

1. Supervisor: Petia Radeva
2. Supervisor: Ricardo Marques
3. Supervisor: Ahmad AlMughrabi

Barcelona, June 10, 2024

Abstract

Food segmentation is crucial in various research fields, such as health, agriculture, and food biotechnology. Segmenting and tracking different types of food in images or videos is undeniably a significant achievement, and it is currently considered a newly emerging topic in today's society. Our study aims to find and develop a production-grade framework for segmenting and tracking various types of food in a given set of images or videos at high-quality performance and near-real-time speed with minimum hardware resources. This unlocks many challenges in real-world applications, such as food volume estimation, calorie estimation, 3D reconstruction, augmented and virtual reality, or digital twins. We introduce FoodMem, a novel framework for segmenting food in 360° scenes. Our framework can effectively segment food portions in a given video and generate accurate masks. Most semantic segmentation models, especially for food-related tasks, have limitations that affect their performance, such as handling different camera locations that did not exist in the training set. Plus, the inference speed for individual images does not fit real-world applications, especially those that focus on video processing. In contrast, memory-based models are becoming popular in object-tracking applications because of their performance and speed. Still, they are limited since they rely on user input, such as the user drawing the input mask manually, which indicates a lack of automation. To overcome these limitations, we propose FoodMem, a novel food video segmentation framework that combines the (1) SETR model to generate segment one- or few- masks of the food portions in a given scene and (2) XMem++, a memory-based tracking model, to track the food masks in complex scenes. Our framework performs better than the state-of-the-art food segmentation frameworks in segmenting food in different camera-capturing locations, illumination, reflection, scene complexity, and food diversity, achieving significant segmentation noise reduction, artifact elimination, and completing the missing parts. We also introduce an annotated food dataset, which covers new challenging use cases not found in previous benchmarks. We conduct extensive experiments on Nutrition5k and Vegetables & Fruits datasets, showing that FoodMem improves the state-of-the-art by 2.5% mean average precision in food video segmentation. Moreover, FoodMem is 58 times faster than the state-of-the-art on average for both datasets. The source code is accessible at: [1].

Resumen

La segmentación de alimentos es fundamental en diversos campos de investigación, como la salud, la agricultura y la biotecnología alimentaria. Segmentar y rastrear diferentes tipos de alimentos en imágenes o videos es un logro considerable y actualmente se considera un tema emergente en la sociedad. Nuestro estudio pretende segmentar y rastrear diferentes tipos de alimentos en un conjunto de imágenes o videos, con un alto rendimiento y velocidad casi en tiempo real, utilizando recur-

[1] https://amughrabi.github.io/foodmem

sos de hardware mínimos. Esto abre numerosos desafíos en aplicaciones del mundo real, como la estimación del volumen de alimentos, la estimación de calorías, la reconstrucción en 3D, la realidad aumentada y virtual, o los gemelos digitales. Presentamos FoodMem, un nuevo marco de trabajo para segmentar alimentos en escenas de 360°. Nuestro marco puede segmentar eficazmente porciones de alimentos en un video dado y generar máscaras precisas. La mayoría de los modelos de segmentación semántica, especialmente aquellos relacionados con alimentos, tienen limitaciones que afectan su rendimiento, como el manejo de diferentes ubicaciones de cámaras que no estaban presentes en el conjunto de entrenamiento. Además, la velocidad de inferencia para imágenes individuales no se adapta bien a aplicaciones del mundo real, especialmente aquellas que se centran en el procesamiento de videos. En cambio, los modelos basados en memoria están ganando popularidad en aplicaciones de seguimiento de objetos debido a su rendimiento y velocidad. Sin embargo, están limitados porque dependen de la intervención del usuario, como dibujar manualmente la máscara de entrada lo que indica una falta de automatización. Para superar estas limitaciones, proponemos FoodMEM, un nuevo marco de segmentación de videos de alimentos que combina (1) el modelo SETR para generar una o pocas máscaras de las porciones de alimentos en una escena dada y (2) XMem++, un modelo de seguimiento basado en memoria, para rastrear las máscaras de alimentos en escenas complejas. Nuestro marco supera a los marcos de segmentación de alimentos más avanzados en la segmentación de comidas en diferentes ubicaciones de captura de cámara, iluminación, reflejo, complejidad de la escena y diversidad de alimentos, logrando una reducción significativa del ruido de segmentación, la eliminación de artefactos y la completación de las partes faltantes. También presentamos un conjunto de datos anotados de alimentos, que cubre nuevos casos de uso desafiantes no encontrados en "benchmarks" anteriores. Realizamos extensos experimentos en los conjuntos de datos Nutrition5k y Vegetables & Fruits, demostrando que FoodMem mejora el estado del arte en un 2.5% en precisión media promedio en la segmentación de videos de alimentos. Además, FoodMem es 58 veces más rápido que el estado del arte en promedio para ambos conjuntos de datos. El código fuente está disponible en: [1].

Resum

La segmentació d'aliments és crucial en diversos camps de recerca, com la salut, l'agricultura i la biotecnologia alimentària. Segmentar i seguir diferents tipus d'aliments en imatges o vídeos és un assoliment significatiu i actualment es considera un tema emergent a la societat. El nostre estudi té com a objectiu segmentar i rastrejar diferents tipus d'aliments en un conjunt d'imatges o vídeos, amb un alt rendiment i velocitat gairebé en temps real, utilitzant recursos mínims de maquinari. Això planteja nombrosos desafiaments en aplicacions del món real, com l'estimació del volum d'aliments, l'estimació de calories, la reconstrucció en 3D, la realitat augmentada i virtual, o els bessons digitals. Presentem FoodMem, un nou marc per segmentar aliments en escenes de 360°. El nostre sistema pot segmentar de manera eficient porcions d'aliments en un vídeo determinat i generar màscares precises. La

majoria dels models de segmentació semàntica, especialment els relacionats amb aliments, tenen limitacions que afecten el seu rendiment, com la gestió de diferents ubicacions de càmeres que no estaven presents en el conjunt d'entrenament. A més, la velocitat d'inferència per a imatges individuals no s'adapta bé a aplicacions del món real, especialment aquelles que se centren en el processament de vídeos. En canvi, els models basats en memòria estan guanyant popularitat en aplicacions de seguiment d'objectes a causa del seu rendiment i velocitat. No obstant això, estan limitats perquè depenen de la intervenció de l'usuari, com dibuixar manualment la màscara d'entrada, la qual cosa indica una manca d'automatització. Per superar aquestes limitacions, proposem FoodMem, un nou marc de segmentació de vídeos d'aliments que combina (1) el model SETR per generar una o poques màscares de les porcions d'aliments en una escena determinada i (2) XMem++, un model de seguiment basat en memòria, per rastrejar les màscares d'aliments en escenes complexes. El nostre sistema supera els marcs de segmentació d'aliments més avançats en la segmentació de menjars en diferents ubicacions de captura de càmera, il·luminació, reflexos, complexitat de l'escena i diversitat d'aliments, aconseguint una reducció significativa del soroll de segmentació, l'eliminació d'artefactes i la completació de les parts mancants. També presentem un conjunt de dades anotades d'aliments, que cobreix nous casos d'ús desafiants no trobats en "benchmarks" anteriors. Duem a terme extensos experiments en els conjunts de dades Nutrition5k i Vegetables & Fruits, demostrant que FoodMem millora l'estat de l'art en un 2.5% en precisió mitjana en la segmentació de vídeos d'aliments. A més, FoodMem és 58 vegades més ràpid que l'estat de l'art en mitjana per a ambdós conjunts de dades. El codi font està disponible a: [1].

Acknowledgements

I would like to express my gratitude to my supervisors, Petia, Ricardo, and Ahmad, for trusting me and giving me the opportunity to work on such an interesting, innovative, and impactful project. Their support and encouragement have been invaluable.

I especially want to acknowledge the unwavering support I have received from Ahmad throughout this entire process. He has been by my side every step of the way, providing guidance, feedback, and attention to my work.

I would also like to extend my thanks to my mother, Esperanza, for her support throughout the project.

Contents

1 Introduction

1.1 Motivation

1.1.1 The Food Volume Estimation project

The Food Volume Estimation project is focused on the significance of estimating the volume of objects in computer vision for measuring and analyzing real-world items. This technology is essential for estimating food volume, which is vital for dietary assessment, meal planning, and culinary automation. The accuracy of food volume estimation is crucial for tracking nutrition and creating personalized food plans for health and well-being. Furthermore, this technology drives advancements in culinary automation, making cooking processes more efficient and inspiring new gastronomic innovations. The Food Volume Estimation project presents a new framework for creating 3D models and accurately estimating object volume from overlapping images. This advanced capability enhances the precision of dietary assessment and meal planning, significantly impacting culinary automation and practical applications in computer vision. The Food Volume Estimation project is also significant to me because it involves the efficient and quick resolution of food semantic segmentation, a challenge our framework can adeptly address. Our framework is a powerful tool for precise food image segmentation, enabling greater accuracy in food volume estimation.

1.1.2 European Sustainable Development Goals

The proposed research project fits 4 of the sustained development objectives of the 2030 agenda:

- "Objective 2: Zero hunger" by reducing food waste and optimizing resource allocation in food distribution and management systems.
- "Objective 3: Ensure healthy lives and promote well-being for all at all ages" by providing advanced tools for food volume estimation with direct application in health.
- "Objective 9: Build resilient infrastructure, promote sustainable industrialization and encourage innovation" by proposing more efficient (and thus less energy-demanding) AI-based algorithms for food volume estimation.
- "Objective 11: Sustainable cities and communities" by fostering urban agriculture, food production, and waste management initiatives.

1.1.3 Personal motivation

My motivation for developing FoodMem comes from my desire to help people live healthier and better lives through the use of technology and computer vision. Having a look at the current food segmentation models [1], I have noticed so many

limitations, and I have seen the opportunity to develop a more specific and efficient solution that can operate in nearly real-time. In addition, I am interested in education and creating awareness about the importance of healthy eating, so I hope that FoodMem will be a helpful tool in this area. Moreover, the possibility of positively affecting the food industry motivates me. Last but not least, my objective is to apply my technical skills to design an innovative solution that improves people's lives.

1.2 Problem definition

Food segmentation, the process of identifying and delineating different food items within images or videos, is an essential aspect of modern computer vision research. Its application spans various fields, from healthcare and nutrition to agriculture and culinary arts. In this thesis, we explore the development and application of a novel framework for food segmentation, addressing key challenges and advancing the state-of-the-art in this critical area of study. Despite its importance, food segmentation poses several challenges, including variability in food appearance, lighting conditions, and camera perspectives.

Different models are available to detect and segment objects in images, such as FoodSAM [2], DEVA [3], kMean++ [4], Detectron2 [5], SAM [6], and YOLOv9 [7]. In contrast, many of these models work on a single image or are not designed specifically for food segmentation, often leading to poor results. For instance, FoodSAM fails to generalize the segmentation across multiple frames in a given video; for example, the estimated mask IDs (i.e., mask colours) are assigned differently for the same food portion in different frames for the same scene. Moreover, FoodSAM fails to segment perfectly and generates masks from different camera views, such as missing food parts, and segments non-food objects, such as plates or tables. Furthermore, FoodSAM is slow, which makes it an unsuitable scene segmentation framework for production scenarios.

To overcome these limitations, our framework can segment food into a series of images and videos. It is designed to deliver high levels of accuracy in near real-time, representing a significant advancement in the field of computer vision for food. Importantly, our framework has applications in various food-related fields.

- **Improved health and nutrition:** Our framework can help individuals monitor their meal consumption, track their diet, manage their weight, and assist those living with conditions such as diabetes. Food segmentation allows people to identify healthier food options and better understand their dietary habits.
- **Enhanced food safety:** Our framework can be applied to implement a novel approach to quality control across various food production and processing sectors. For instance, our framework can create masks that detect contaminants and allergens, ensuring that food products adhere to established safety standards and mitigate health risks for consumers.

- **Reducing food waste:** Our framework can identify where food is being wasted by accurately tracking food from production to consumption. This valuable information can be used to develop strategies that minimize waste, helping to protect resources and promote sustainability.
- **Educational tool:** Our framework can serve as a powerful educational tool by increasing awareness of healthy eating. The application can be utilized in schools and for public health campaigns to educate users about nutrition, food safety, and the environmental impact of food choices.
- **Metaverse and gaming experience:** Our framework can improve the realism and detail of cooking and farming simulation games by incorporating it into viewing synthesis and 3D reconstruction algorithms. This will create more engaging and lifelike games for players. Introducing food segmentation will enhance the realism of the gameplay and enhance the overall entertainment experience.

1.3 Bounded and unbounded scenes

In the context of our research, it is important to distinguish between bounded and unbounded scenes, as these terms describe the movement constraints and operational environments of the camera used for capturing images or videos. Our framework outperforms state-of-the-art methods at segmenting food objects in unbounded scenes, which addresses more complex scenes and applies to many more situations.

Bounded scenes refer to environments where the camera follows a predefined path or trajectory. This controlled motion ensures systematic coverage of the scene, which is crucial for applications requiring consistent and repeatable imaging conditions. The predefined path helps maintain uniformity in image capture, making it easier to analyze changes over time, detect specific objects, or monitor particular areas accurately. On the other hand, unbounded scenes are characterized by free or unrestricted camera movements. These scenarios typically involve handheld or mobile cameras. Unbounded scenes present a greater challenge for image analysis due to the variability in camera angles, distances, and motion dynamics. This variability can lead to inconsistent image capture conditions, making it more difficult to segment and track objects accurately. An example of a project that deals with such challenges is MiP-NeRF 360 [8], which addresses neural radiance fields in 360° scenes with unrestricted camera movement, demonstrating advanced techniques to handle the complexities of unbounded environments effectively.

1.4 Objectives

Our study aims to apply academic knowledge and practical skills to develop a production-grade food video segmentation. The objectives are directed toward applying the knowledge learned at the university, improving programming skills, de-

veloping the program, so it is versatile and could be applied anywhere in the world related to food, learning techniques used in segmentation and tracking, achieving near real-time execution, and expanding knowledge in the area of computer vision.

- **Explore, understand and deep-dive into the computer vision problems related to image segmentation and tracking:** Our main goal is to study and comprehend segmentation and tracking techniques [9], particularly in the context of food imagery. This may involve examining current models and methods and identifying connections between techniques, for example, using FoodSAM for segmentation and XMem++ for tracking.
- **Design and validate a new flexible and globally applicable algorithm for food segmentation:** Our goal is to create a versatile program that can be used for various applications worldwide. Additionally, the program should be designed to evolve and adapt to new emerging needs and technologies, such as panoptic segmentation, volume estimation, and calorie estimation.
- **Achieve nearly real-time execution of food segmentation and tracking:** Our goal is to meet program expectations by achieving near-real-time performance in segmentation and tracking results. This involves optimizing algorithms, data processing pipeline, and hardware utilization to reduce execution latency and enhance user experience.
- **Apply university knowledge:** Our goal is to apply theoretical and practical knowledge acquired during university studies in subjects such as computer vision, image processing, and software engineering. This includes understanding the fundamental concepts, algorithms, and methodologies related to the development of our framework.
- **Expand knowledge in computer vision:** Our goal is to expand knowledge and experience in the field of computer vision beyond the academic curriculum. This involves researching state-of-the-art methodologies and applications to enhance our framework capabilities and contribute to the growth of the field.
- **Improve knowledge of Python and relevant libraries:** Our study aims to gain a deeper understanding of programming languages such as Python and related libraries, including NumPy, PyTorch, and scikit-learn. This involves hands-on applications and experiments to grasp the functionality of these tools.

1.5 Contributions

Our framework is a significant advancement in segmentation. It provides substantial improvements and outperforms existing state-of-the-art methods. To overcome the limitations in Sec. 2, our contributions are listed below:

1. We build a novel near-real-time food segmentation architecture for videos that combines SETR [10] and XMem++ frameworks as a first exploration for food video segmentation.

2. We introduce a novel dataset tailored for food image segmentation tasks. Our dataset comprises a comprehensive selection of dishes sourced from the Nutrition5k [11] dataset, encompassing 31 distinct dishes with a total of 1356 annotated frames. Additionally, we include 11 dishes from the Vegetable & Fruits [12] dataset, augmented with 2308 annotated frames. Our dataset features 42 diverse dishes, accompanied by 3664 meticulously annotated frames. We believe this expansive dataset is a valuable resource for advancing research in video food segmentation and related computer vision tasks.

3. We conducted an extensive series of experiments on our dataset to assess the effectiveness and flexibility of our framework against the baselines (we reproduced the results).

4. Our framework outperforms the state-of-the-art performance in video food segmentation for 2.5% mean average precision with similar recall.

5. Our framework is 58 times faster than the baselines' inference time on average for both datasets.

6. Since not all of the baselines are open source, we implemented kMean++ [4] for better benchmarking.

1.6 Document Structure

The remaining part of this thesis structure can be used as follows:

1. **State of the art** (Sec. 2): We review and summarize existing literature, research gaps, and methodologies relevant to the topic. We discuss previous studies, approaches, and technologies related to food segmentation, tracking, and computer vision. The aim is to provide context for the thesis and identify areas for further exploration.

2. **FoodMem Methodology** (Sec. 3): We present our proposal included in the thesis project covers our framework subsection, describing the architecture and functionality of the food segmentation and tracking system. It also explains dataset creation, detailing how the dataset was collected and annotated.

3. **FoodMem Implementation** (Sec. 4): We explore the technological and software engineering aspects that were used to develop our framework.

4. **Validation** (Sec. 5): We demonstrate our methodology in practice and present and discuss the results. We describe the experimental setup, including the

dataset and evaluation metrics. We report the results of experiments conducted with our framework, presenting quantitative performance metrics and qualitative observations. Moreover, we compare our findings with existing baselines. We also present our framework modules' contributions in different settings. It involves modifying the number of initial masks that the SETR module must generate so that XMem++ takes into account when segmenting the rest of the video. Finally, we list the limitations of our framework, this aims to provide transparency regarding our framework capabilities and improvement areas.

5. **Conclusions and future work** (Sec. 6): We present a summary of the key findings from our framework. Moreover, we also provide insights for future research and development, outlining areas where the model can be expanded.

6. **Appendix** (Sec. 7): Additional information of our framework. It aims to provide more images and results.

2 State of the art

In this section, we examine the methods and studies related to food segmentation and tracking in the field of computer vision [13]. This review aims to provide context for our thesis by examining previous works that utilize the latest techniques and advancements in this area. We analyze past projects to identify patterns, standard practices, and new technologies used in food segmentation and tracking systems. Furthermore, we outline the limitations and difficulties inherent in the current methods, underscoring the necessity for the creation of our framework. More precisely, this section helps position our framework within the wider research context, demonstrating how our framework contributes to the field by tackling specific deficiencies and introducing new methodologies. In the following subsections, we explain the overview, model and limitations of the methods and studies related to food segmentation and tracking, such as Vision transformers (see Sec. 2.1), SAM (see Sec. 2.2), FoodSAM (see Sec. 2.3), DEVA (see Sec. 2.4), XMem (see Sec. 2.5), XMem++ (see Sec. 2.6) and kMean++ (see Sec. 2.7).

2.1 Vision transformer (ViT)

The Vision Transformer (ViT) [14] is a type of neural network architecture primarily designed for image classification tasks that utilize self-attention mechanisms. The concept behind ViTs is to treat an image as a sequence of patches. These patches are linearly embedded and then fed into a transformer architecture consisting of multiple layers of self-attention and feedforward neural networks. This approach enables ViTs to capture long-range dependencies in images and learn representations that are insensitive to translations, rotations, and other transformations.

2.1.1 Model

The model architecture of a ViT comprises several key components enabling it to process and extract meaningful information from images. These are the main elements of a ViT model, as shown in Fig. 1:

- **Patch embedding:** The input image is divided into fixed-size, non-overlapping patches. Each patch is then transformed into a lower-dimensional vector space to create initial token embeddings. These embeddings get the information from the image patches and are used as input for the transformer encoder.
- **Transformer encoder:** The patch embeddings are processed by a series of transformer encoder layers.
- **Positional embedding:** Since ViT does not have convolutional layers that inherently capture spatial information, positional encodings are added to the patch embeddings to provide information about the spatial location of each patch within the image.

- **MLP Head:** The transformer encoder produces a series of token embeddings that represent the image. A classification head is then included on these embeddings to generate the final classification predictions.

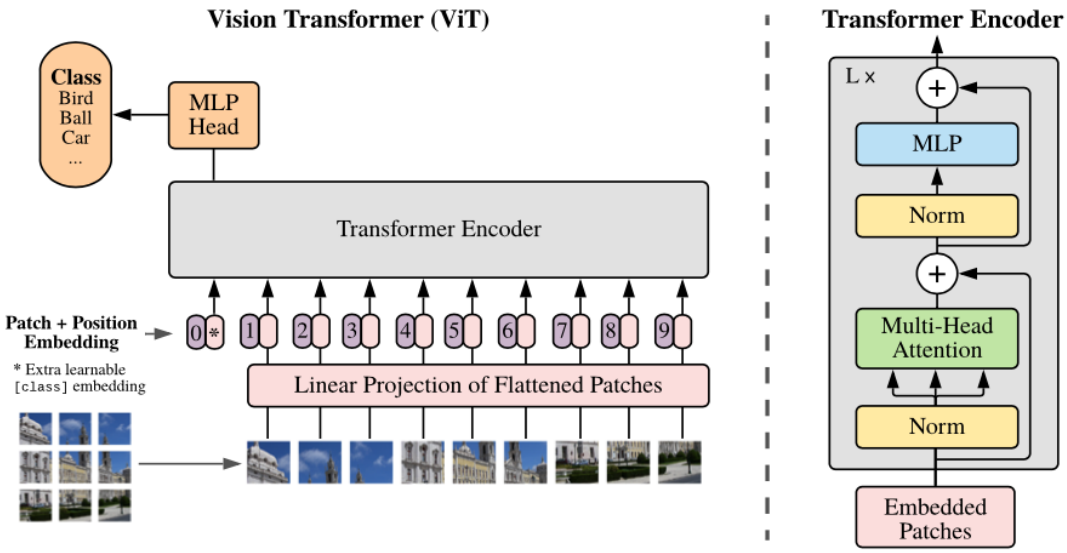


Figure 1: ViT model design, from [14]. The input image is split into fixed-size patches, linearly embedded each of them, adds position embeddings, and feeds the resulting sequence of vectors to a standard Transformer encoder. To classify, an extra learnable classification token is added to the sequence.

2.1.2 Limitations

Vision Transformers have shown impressive performance in various image processing tasks. However, they also have some limitations. Firstly, ViTs often require significant memory resources due to the quadratic complexity of self-attention mechanisms, especially when dealing with large input images or long sequences of tokens. Secondly, despite efforts to integrate positional encodings, ViTs may struggle to capture precise spatial information important for tasks like localization or object detection. Additionally, ViTs rely on fixed-size patches extracted from input images, limiting their adaptability to images of varying resolutions and potentially leading to information loss or distortion. Moreover, training large ViT models can be computationally expensive and time-consuming, requiring significant computational resources and longer training times. Lastly, ViT performance can be sensitive to parameters like patch size and grid layout during pre-processing, making it challenging to find optimal configurations for different datasets or tasks.

2.2 The Segment Anything Model (SAM)

The Segment Anything Model (SAM) [6] is a state-of-the-art model for segmentation tasks in computer vision. Developed by Meta AI, SAM aims to provide a highly

versatile and generalizable solution for segmenting objects and regions of an image without the need for specific object labels during training.

2.2.1 Model

SAM addresses the issue of promptable segmentation tasks and real-world application requirements. Specifically, SAM must support flexible prompts, compute masks in near real-time for interactive use, and effectively handle ambiguity. The model is designed to tackle the challenges of promptable segmentation tasks and the requirements for real-world applications. A simple and effective design consists of these components, as shown in Fig. 2:

1. **Image encoder:** It is a Vision Transformer (ViT) (see Sec. 2.1) pre-trained with Masked Auto-encoders [15] to process high-resolution inputs. The image encoder processes the input image to generate an image embedding which captures essential visual features and is computed once per image, allowing it to be reused with different prompts to optimize efficiency.

2. **Prompt encoder:**

   - **Types of prompts:** SAM supports different types of prompts, including sparse prompts like points, boxes, and text, as well as dense prompts like masks.
   - **Encoding sparse prompts:** Points and boxes are represented using positional encodings combined with learned embeddings for each prompt type. Free-form text prompts are encoded using a text encoder from CLIP [15].
   - **Encoding dense prompts:** Masks are applied using convolutional layers and then combined with the image embedding using element-wise operations.

3. **Mask decoder:**

   - **Function:** The mask decoder utilizes the image embedding, prompt embeddings, and output token to make predictions for the segmentation mask.
   - **Architecture:** The mask decoder is inspired by transformer decoder blocks and uses self-attention and cross-attention mechanisms between the prompts and the image embedding. This interaction updates the embeddings and the final output token is passed through a Multi-Layer Perceptron (MLP) to predict the mask's probability at each location in the image.

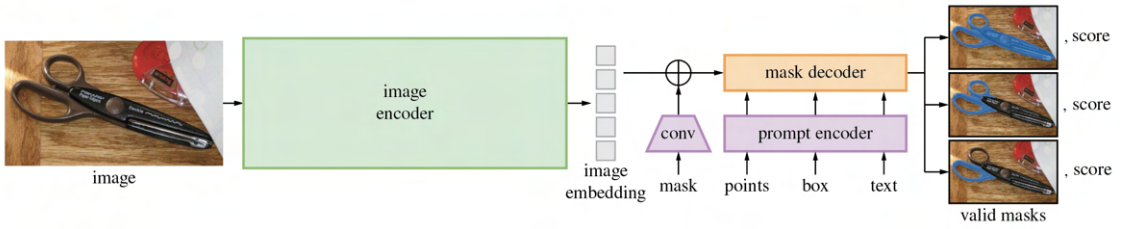


Figure 2: SAM model design, from [16]. SAM model accepts a single image as an input, which enters into the encoder (ViT) to generate an embedding. User prompts, such as points, boxes, and masks, are additional inputs to SAM models, which enter the prompt encoder. Later, the mask decoder uses the image embedding and the prompt embedding to generate the final masks.

2.2.2 SAM Training

SAM uses a combination of focal loss [17] and Dice loss to supervise mask prediction, address class imbalance, and improve segmentation boundary accuracy. The model is trained for promptable segmentation tasks using various geometric prompts. The training simulates an interactive setup, sampling prompts randomly over multiple rounds, which helps SAM integrate well into interactive systems.

2.2.3 Main characteristics of SAM

Below are the main characteristics of SAM [6]:

- **Generalization:** SAM has the ability to generalize across various objects and scenes. Unlike traditional segmentation models that are trained on specific tasks or objects, SAM can segment almost anything in any image, making it flexible and widely applicable.
- **High-quality segmentation:** SAM is capable of generating high-quality segmentation masks, even for objects or regions not encountered during training. This is accomplished by leveraging state-of-the-art deep learning techniques and training on diverse datasets.
- **Scalability:** SAM is designed for scalability and can handle large, complex datasets. It excels in a wide range of image types and excels in segmentation tasks, from simple objects to complex scenes.
- **Promptable interface:** SAM has a unique feature, its promptable interface. This system lets users interactively specify the segmentation they want by using different prompts, like points, boxes, or text descriptions. This means that SAM can be used for segmentation tasks without needing specific training for those tasks.

2.2.4 Limitations

SAM often misses fine structures and incorrectly creates small disconnected components [18]. This can result in inaccurate representations of complex objects and undesired artifacts in the segmentation. SAM's general nature also requires extensive computations to address all image potentialities and ambiguities. Yet, SAM is designed for single-image processing, so its segmentation capabilities are optimized for a single image rather than video data. Considering SAM's versatility, these limitations might make it unsuitable for our specific objectives.

2.3 FoodSAM

FoodSAM is a new framework designed specifically to address the challenges of food image segmentation. It was created because SAM has limitations in capturing specific class information in the generated masks. This compromises the quality of results, especially when dealing with diverse food appearances and imbalanced ingredient categories. FoodSAM aims to improve the segmentation process by combining SAM-generated masks with original semantic masks, enhancing segmentation quality while preserving category information. FoodSAM also presents methodologies for object detection to identify non-food objects existing in the images. Through object detection combined with semantic segmentation methodologies, FoodSAM can perform panoptic segmentation [19] on food images, including segmentations at multiple granularity levels.

2.3.1 Model

FoodSAM, as shown in Fig. 3, consists of three main components: SAM, which identifies objects in images; a semantic segmentation module, which labels the objects; and an object detector, which recognizes non-food items. The process begins by improving semantic segmentation by combining the output from SAM with semantic labels to enhance object identification. Subsequently, it conducts semantic-to-instance segmentation, which treats each object individually and merges related masks to identify separate ingredients. In the instance-to-panoptic segmentation stage, non-food items are identified separately using the object detector and combined with instance masks to provide a complete scene overview. Furthermore, FoodSAM allows interactive segmentation based on user prompts, offering flexible and interactive food and non-food object segmentation at various levels of detail.

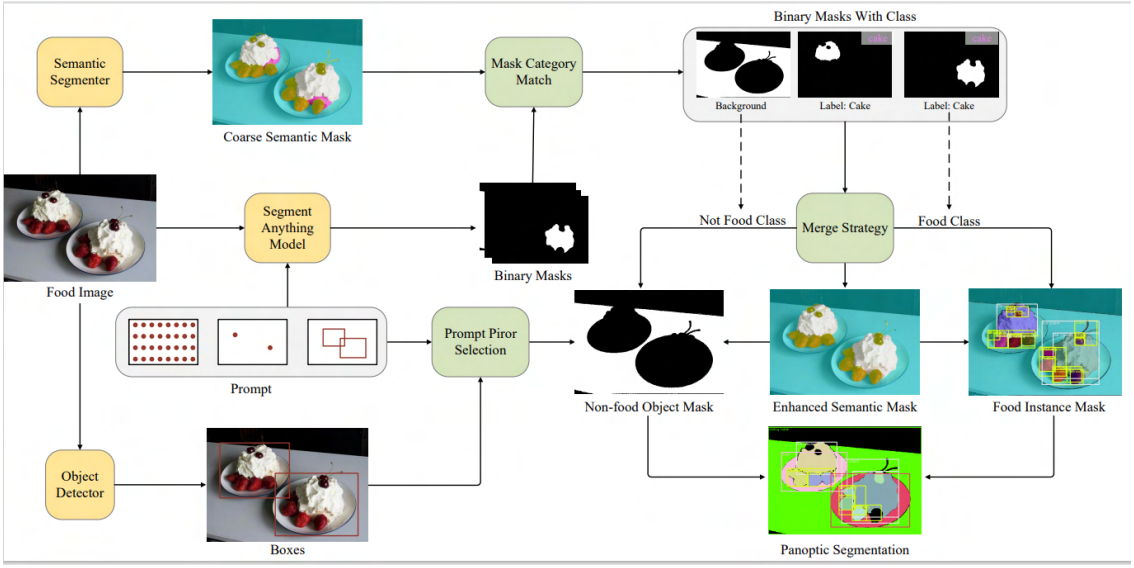


Figure 3: FoodSAM model design, from [2]. The FoodSAM model accepts a single image as an input. Firstly, the image goes through the semantic segmenter module, which labels the objects. Secondly, the image goes through SAM, which identifies the objects. Thirdly, once both modules are finished, masks generated by the modules are matched to enhance the identification of food in the image. Fourthly, the image goes through the object detector, which recognizes non-food items. Lastly, the matched and object detector masks are merged to enhance masks, generate an improved semantic mask, and realize a panoptic segmentation mask.

2.3.2 Limitations

FoodSAM is highly effective for processing individual images, but it lacks the capability to handle videos directly. As a result, each frame must be segmented separately when working with videos, significantly increasing processing time. Additionally, unlike some video processing models, FoodSAM does not remember previous frames, which further contributes to the extended processing time and makes it impractical for real-time or efficient video segmentation tasks. Furthermore, FoodSAM saves all generated masks from SAM and FoodSAM onto the disk, leading to additional computational overhead, particularly when dealing with numerous frames in a video, which affects the efficiency and practicality of using FoodSAM for video segmentation.

2.4 DEVA

Tracking Anything with Decoupled Video Segmentation (DEVA) is the state-of-the-art computer vision method for tracking objects in videos. Unlike traditional methods, DEVA separates the process of segmenting objects in each frame from the tracking process. It starts by precisely identifying and delineating objects in the individual frames through segmentation. Then, it links the segmented objects across frames using an advanced tracking algorithm to track the objects over time

accurately. This decoupled approach allows DEVA to robustly and accurately track objects, even in challenging scenarios with occlusions or changes in object appearances.

2.4.1 Model

DEVA approaches the video segmentation task differently by separating it into two components [3]:

- **Image-level segmentation model:** Trained specifically for the target task, this model generates a task-specific segmentation hypothesis for individual video frames. Each frame's segmentation is represented as a set of separate binary segments.
- **Bi-directional temporal propagation:** Trained on class-agnostic mask propagation datasets, the model propagates segmentation hypotheses by the image segmentation model through the full video, incorporating past segmented frames as memory.

The process, as shown in Fig. 4, begins with the image segmentation model, which provides initial segments for every frame. These segments are then refined using a small clip of future frames to reach a consensus. The temporal propagation model propagates the improved segments to the next frames. This process continues iteratively, with new images' segmentation results periodically added and combined with the propagated segmentation results. These are the two key steps involved in bidirectional propagation:

- **In-clip consensus:** The segments form a small clip of future frames that are aligned spatially, represented as object proposals, and an indicator variable is optimized to select the most appropriate proposals.
- **Merging propagation and consensus:** The segments propagated from the previous frames are combined with the consensus segments of the future frames. This process includes matching segments from both sets and merging the matched pairs while keeping unmatched segments.

Moreover, inactive segments are periodically cleared from memory to reduce computational costs, making video segmentation efficient and effective for many scenarios.

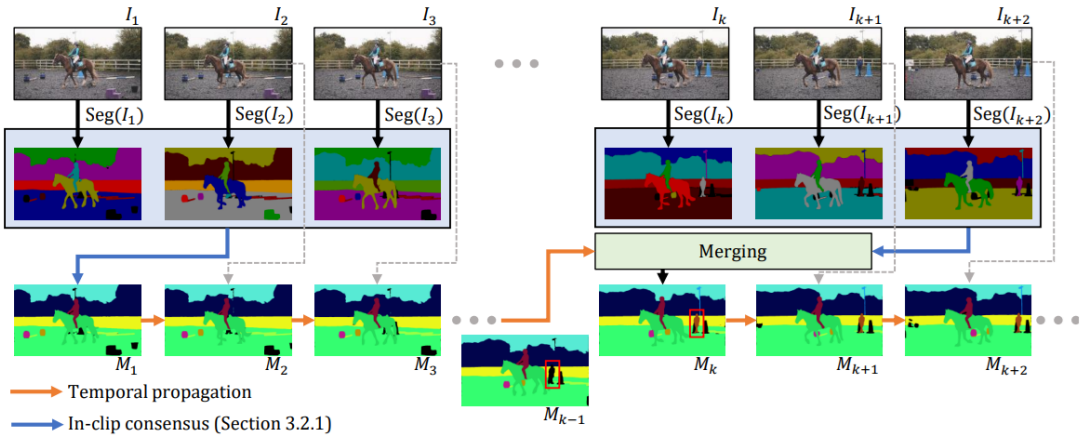


Figure 4: DEVA model design, from [3]. Firstly, image segmentation is filtered with in-clip consensus and temporally propagates this result forward. The propagated results are merged with in-clip consensus to incorporate new image segments.

2.4.2 Limitations

DEVA has several limitations that impact its performance. One major limitation is its inability to independently detect new objects due to its temporal propagation model not being specialized for any particular task. This can delay the detection of new objects, as they are only incorporated into the segmentation during periodic updates with the in-clip consensus. Additionally, DEVA's decoupling approach requires significant computational resources to reason through various possibilities and ambiguities within an image, leading to reduced efficiency compared to end-to-end approaches, especially when large amounts of training data are available. New objects in the scene may also initially be missing from the segmentation and are only detected in subsequent frames, causing delays in inference time based on the frequency of merging with in-clip consensus. Furthermore, DEVA's generic nature means it is not optimized for specific tasks or domains, and additional customization may be necessary for optimal performance in such cases.

2.5 XMem

XMem [20] is a video object segmentation model designed to segment objects of interest across multiple frames in a video. XMem incorporates a memory mechanism inspired by the Atkinson-Shiffrin model [21], which retains information about the segmented objects. Such a memory mechanism also enables the propagation of segmentation annotation from a few annotated frames to other frames in the video, ensuring consistent and accurate segmentation in the sequence.

2.5.1 Model

XMem, as seen in Fig. 5, utilizes the following four components to realize its purpose:

- **Memory mechanism:** The memory module in XMem retains segmented object information over time. This memory enables the model to store the segmentation annotations of a limited number of annotated frames and propagate them to further frames in the video.
- **Propagation of segmentation annotations:** Using the information stored in its memory, XMem propagates segmentation annotations from annotated frames to other frames in the video, ensuring consistent and accurate segmentation throughout the video.
- **Long-term segmentation:** XMem tries to attain long-term segmentation by effectively leveraging the information stored in its memory. It retains the segmentation annotations of the frames over time, providing consistency and accuracy in the object segmentation of multiple frames.
- **Atkinson-Shiffrin model inspiration:** The memory mechanism in XMem draws inspiration from the Atkinson-Shiffrin model, which allows the existence of multiple memory stores, including sensory memory, short-term memory, and long-term memory. XMem adapts this to maintain segmentation information over a video.

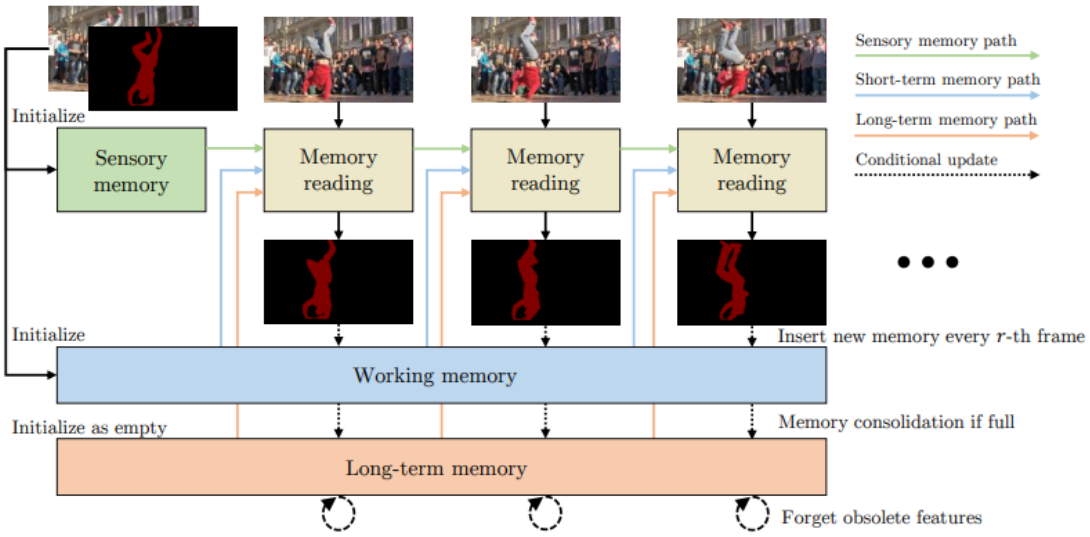


Figure 5: XMem model design overview, from [20]. The memory reading operation extracts relevant features from all three memory stores and uses those features to produce a mask. The sensory memory is updated every frame, while the working memory is only updated every few frames. The working memory is consolidated into the long-term memory in a compact form when it is full, and the long-term memory will forget obsolete features over time.

2.5.2 Limitations

XMem has several limitations that affect its performance. One issue is its sensitivity to fast-moving objects within video frames; even with a fast-updating sensory

memory, the model may struggle to capture and track the motion of these objects, leading to segmentation errors, especially when objects move suddenly. Additionally, XMem faces challenges in situations with significant motion blur, as the precision of object segmentation may be compromised. The model may have difficulty accurately differentiating or isolating objects with blurred edges, resulting in inaccuracies in the segmentation output, particularly in scenes with fast-paced action or camera movements. Furthermore, the memory-based approach XMem uses results in increased computational overhead, particularly when propagating segmentation annotations across video frames. This heightened computational complexity could hinder the model's scalability, especially when processing high-resolution or long-duration videos, and may necessitate significant computational resources for efficient inference. Lastly, while XMem is a versatile tool, its architecture may lack specialized optimizations for specific tasks or domains, potentially resulting in suboptimal performance in situations with unique requirements, where task-specific models might offer better segmentation accuracy and efficiency.

2.6 XMem++

XMem++ [22] is a state-of-the-art video object segmentation model designed to operate efficiently at a production level. It is based on its predecessor, XMem, and aims to deliver high-quality segmentation results for video frames with minimal annotated data. XMem++ refines the memory mechanism and processing efficiency, making it suitable for real-world applications, considering resources and annotated data restrictions.

2.6.1 Model

The model, as noticed in Fig. 6, is based on the XMem architecture and includes a deep convolutional neural network with multiple parts and three types of memory modules:

- **Sensory memory:** Captures information about the motion by processing the changes between consecutive frames.
- **Short-term memory:** Stores recent frame information for quick access.
- **Long-term memory:** Compresses and retains key features from earlier frames to manage memory efficiently and handle longer videos.

Regarding the memory update mechanism, there are two types:

- **Permanent working memory:** Stores ground-truth references (annotated frames) for the whole video. This avoids frame compression or moving to long-term memory, ensuring it strongly affects the segmentation quality.

- **Temporary working memory:** Stores intermediate frames, and their predictions are used and updated frequently during the segmentation process.

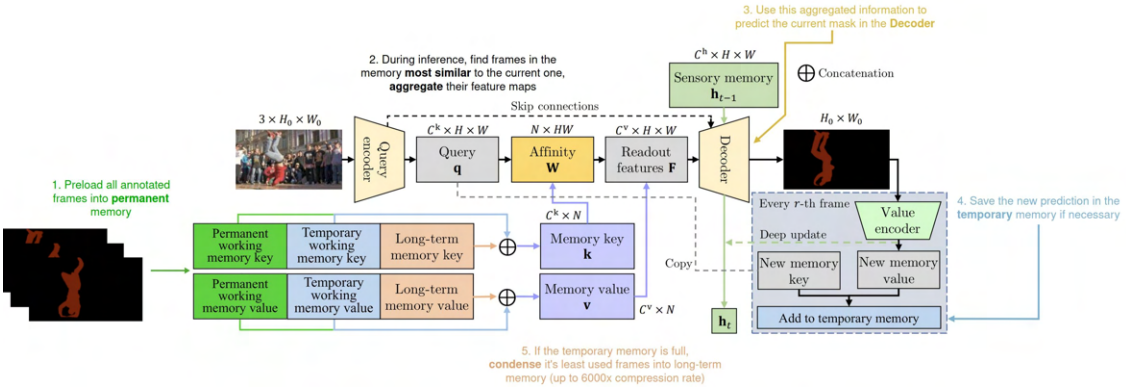


Figure 6: XMem++ architecture, from [22]. Firstly, the model loads all the available training data, including annotated frames, into the permanent memory. Secondly, during the inference, the model looks for memory frames most similar to the current input frame. It then aggregates the feature maps of these similar frames to create a richer representation. Thirdly, the aggregated information from similar frames is input to the decoder module, which uses this information to learn the relationships between frames and predict the mask for the current frame. Fourthly, the model focuses on updating the temporary memory, which serves as a short-term store for recent predictions. Lastly, once the temporary memory is full, the model will condense the least used frames into the long-term permanent memory.

2.6.2 Segmentation process

1. **Frame annotation:** The user annotates a frame, which is processed to generate a dense segmentation mask.

2. **Permanent memory:** The mask is stored in the permanent working memory as a reference for segmenting other frames.

3. **Feature extraction:** Two feature maps are extracted for every frame stored in memory:

    - **Key:** Contains information about the entire frame used for matching similar frames.
    - **Value:** Contains target-specific information for predicting segmentation masks.

4. **Segmentation prediction:** For a new frame, the model searches for similar frames in the memory modules, aggregates information from the matched keys, and predicts the segmentation mask.

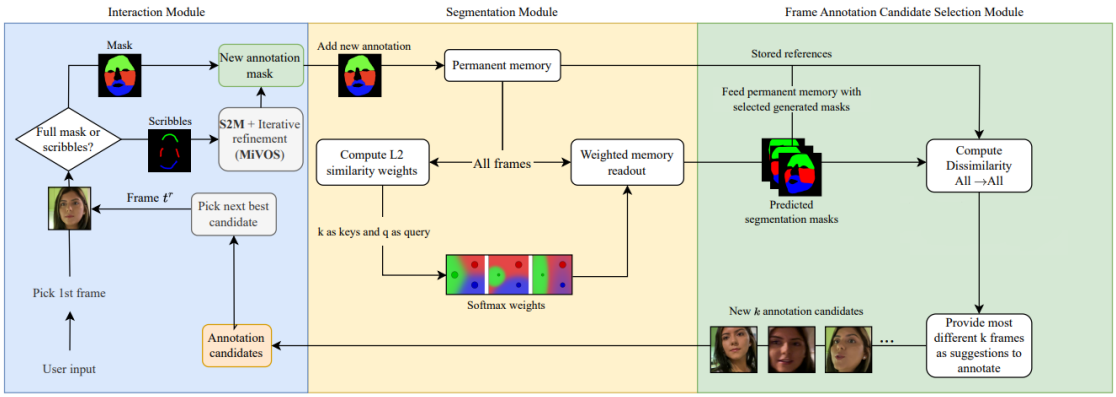


Figure 7: XMem++ interaction flow, from [22]. The user provides initial annotations, segmentation is performed, and then, using predicted masks, new annotation candidates are chosen and given to the user. This loop is repeated until segmentation high quality is achieved.

2.6.3 Improvements over XMem

XMem++ stores annotated frames permanently to ensure they are available during segmentation. This resolves issues with temporary memory, where additional annotated frames might become compressed and lose their impact. Permanent memory also helps create seamless transitions when the target object changes between scenes by providing consistent references throughout the video. Additionally, the model can perform matching based on content rather than the position of the frames within the video, improving segmentation quality and efficiency.

2.6.4 Limitations

Even though XMem++ represents an advancement over its predecessor, XMem, it still exhibits certain limitations [22]. Segmentation quality may suffer for frames blurred by rapid object movements, and the similarity measure used for selecting annotation candidates may also be affected by blur. Moreover, the model can erroneously switch to the wrong target when multiple similar or identical objects are present in the frame, particularly when they occlude each other. Additionally, XMem++ heavily relies on user input for segmentation accuracy, necessitating clicks, masks, or annotations. Furthermore, the method encounters difficulties with highly deformed objects and intricate details, posing challenges for accurate segmentation. Despite its versatility in handling various video segmentation tasks, XMem++'s generic nature may sometimes result in suboptimal performance in highly specialized or extreme scenarios.

2.7 kMean++

kMean++ [4] is an improvement over the classic K-means algorithm of clustering [23] to optimize the selection quality for initial centroids. In the traditional K-means, the initial centroids are randomly picked from the data points. This may result in suboptimal clustering, particularly in high-dimensional space. kMean++ solves this problem through a process of iteration on initial centroid points that are far from each other to increase the possibilities of a solution of global optimality. Since the article describing the kMean++ algorithm is publicly available, we developed our implementation due to the code's unavailability. We needed to implement kMean++ [24] because we required additional methods employing different segmentation techniques for comparison with our framework.

2.7.1 Limitations

kMean++ exhibits several limitations that impact its performance. Firstly, it suffers from low precision, being unable to capture intricate details in images, which often leads to failure to distinguish similar objects or accurately identify detailed boundaries. Secondly, kMean++ lacks contextual understanding or memory incorporation, treating each image independently without considering any surrounding context. This limitation hampers its ability to comprehend complex scenes or objects within a broader context. Lastly, while kMeans++ does not necessitate pre-trained models, relying solely on clustering restricts its capability to effectively segment objects, as it lacks the benefit of prior knowledge or learned features.

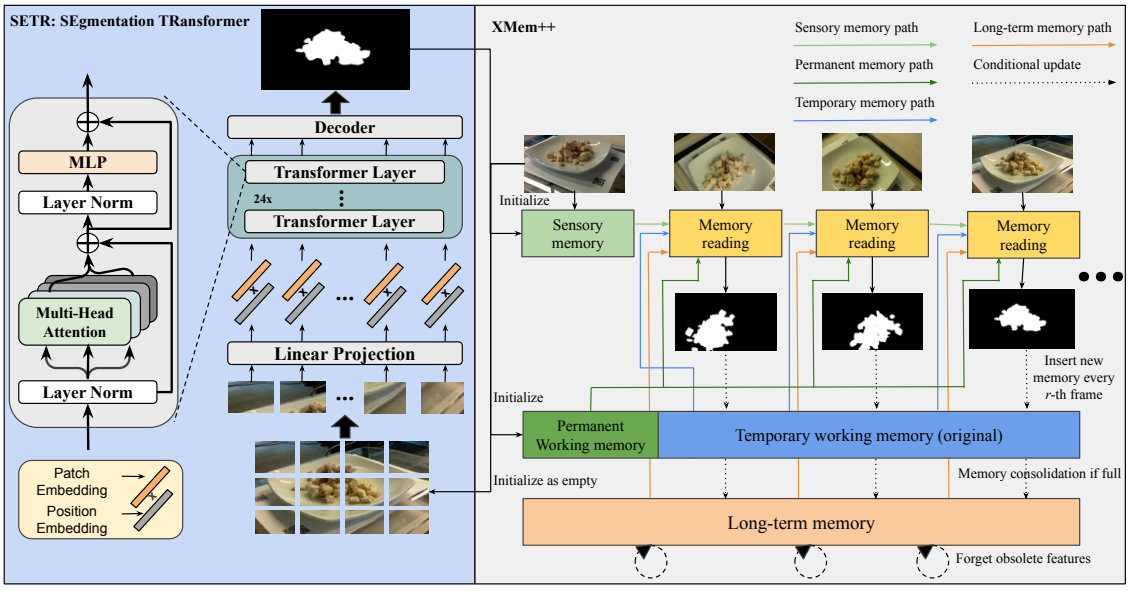
2.8 Research gap

Most of the previous models on semantic segmentation, in particular for food-related applications, have different limitations that affect their effectiveness. More briefly, Table 1 identifies the specific research gaps in state-of-the-art models. This makes FoodMem an even more solid solution for segmentation tasks involving food, increasing its performance and applicability in real scenarios. Table 1 examines various factors influencing the efficacy of existing segmentation models. Firstly, the user's flexibility, denoting the models' capacity to incorporate user preferences or directives. Following this, the efficiency of inference time is evaluated across the models, emphasizing the importance of swiftness. GPUs extensive factor refers to GPU resource consumption among the models, influencing the computational demands of segmentation algorithms. Furthermore, memory-friendly represents if the model relies on a memory system to consider previous frames. Prompts are also examined, facilitating specific instructions or guidance from users. Moreover, the table assesses the generalization capabilities of the models, highlighting their adaptability across different datasets or scenarios. Finally, compatibility with both image and video inputs is examined, reflecting the models' versatility in handling diverse visual data types.

Table 1: A summary of the limitations and research gaps in existing models.

	SAM	**FoodSAM**	**DEVA**	**XMem**	**XMem++**	**kMean++**	**Ours**
User input	✓	✓	✓	✓	✓		
Inference time	✓		✓	✓	✓	✓	✓
GPUs extensive		✓					
Memory-friendly			✓	✓	✓		✓
Prompts	✓	✓	✓				
Generalization	✓		✓	✓	✓	✓	
Image input	✓	✓	✓	✓	✓	✓	✓
Video input			✓	✓	✓		✓

3 FoodMem Methodology



(a) SETR: Generate the first mask

(b) XMem++: Generates masks using a given mask

Figure 8: FoodMem model architecture. We used a single image input for simplicity. Our two-stage framework (a) shows the SETR framework, where it accepts an image and generates a mask, followed by (b) XMem++, which accepts the mask and a set of images as a given input and produces masks for all frames.

3.1 Overview

Our proposed model offers a new automated method of segmenting the food portion per scene and incorporating a per-scene segmentation using the SETR [10] model that produces one or a few food masks, followed by XMem++ [22], the memory-based model, to track food masks in the complex scene.

More briefly, in the beginning, our idea was to create a tool for segmenting food (see Sec. 3.2). However, we soon realized that the time complexity of FoodSAM program was problematic (see Sec. 2.3.2). We focused on optimizing the algorithm to find the right solution to address this. Based on our findings, we discovered that we could develop a program capable of quickly and efficiently segmenting videos, thus solving the time issue. To accomplish this, we relied on two existing algorithms, SETR and XMem++, to create our framework. Our strategy involved using SETR to perform segmentation in the first frame of the video and then passing this mask to XMem++ to segment the rest of the video. It is important to note that XMem++ has the ability to track segmented objects throughout the video. We incorporated a memory feature into our framework to track all the foods in the video. This way, we can overcome the limitations of FoodSAM, as its baseline is SETR, and XMem++ by effectively combining both algorithms.

3.2 Our research question

Our research question for this project is: "If I have a set of frames for different food objects, can I track and segment them uniquely across all the frames?" This is the base for our goal: develop a tool to precisely segment and track food objects across video frames.

3.3 Preliminaries: SETR

SEgmentation **TR**ansformers (SETR) [10] is an architecture created for carrying out image segmentation tasks in computer vision. It combines transformer architectures with the specific needs of image segmentation. SETR works by dividing an input image into embeddings. These embeddings are then processed through a series of transformer layers, allowing the model to capture local and global dependencies within the image. This enables SETR to comprehend the spatial relationship between different parts of the image and make predictions about objects' presence and boundaries, as shown in Fig. 8 (a).

SETR has been trained on the FoodSeg103 [25] dataset, which is a recently introduced dataset for food image segmentation tasks. It includes 7118 images showing 730 different dishes. The train set includes 4983 images with 29530 ingredient masks, while the test set includes 2135 images with 12567 ingredient masks.

3.4 FoodMem

Our framework accepts a set of images and produces a corresponding set of masks. In Sec. 3.4.1, we explain our Command Line Interface (CLI) and how to use our framework; in Sec. 3.4.2, we explain the workflow of our framework; and finally, in Sec. 3.4.3 we describe the expected output of our framework. In Sec. 3.5, we describe in detail our proposed dataset.

3.4.1 CLI

Our framework's input is specified using a set of command-line arguments, which define the paths to the video, masks, and output directory. There are also optional flags for visualization, timing, and specifying the number of masks to be generated by SETR. Here is a detailed explanation of each input:

Table 2: Input flag options, where we can see flag types, routes, constraints, and descriptions.

Flags				
	Type	**Route**	**Constraints**	**Description**
video	String	Video frames directory	Frames must be ('0.jpg', '1.jpg', ...) Frames must be in JPG format	Reads video frames
masks	String	Masks directory	Masks must be ('0.png', '1.png', ...) Masks must be in PNG format	Reads and saves masks
output	String	Masks directory	Output path folder is created if it does not exist	Saves the results
show_vis	Boolean	On screen		Saves the predicted SETR mask overlapped to the corresponding frame
show_overlay	Boolean	On screen		Saves the predicted FoodMEM masks overlapped to their corresponding frames
show_time	Boolean	On screen		Displays the execution time
num_masks	Int	On console	The number of masks must be between 0 and the total amount of frames	Indicates how many masks will SETR generate

3.4.2 Workflow

Our framework combines the models of SETR and XMem++ to achieve efficiency and speed in semantic segmentation in videos. SETR initially segments a single image, while XMem++ uses this information to track and segment objects in the video by remembering and tracking everything that has appeared in the scene. The design model for our framework is shown in Fig. 8.

In Sec. 3.4.2.1, we detail SETR, explaining its implementation and contribution. In Sec. 3.4.2.2, we detail XMem++, how it integrates with SETR, and how it uses its memory capability to realize consistent and accurate segmentation across the video.

3.4.2.1 SETR

FoodSAM comprises three models: SAM, semantic segmenter (i.e. SETR), and object detector. SAM generates multiple independent binary class masks, the semantic segmenter provides food category labels by matching mask categories, and the object detector provides non-food classes for background masks. Since we aim to segment food simply, we have completely isolated the semantic segmenter. This significantly speeds up execution by removing two models that do not influence our purpose. Upon running the code, the first step is to identify the first frame of the video, except if more masks to be generated were introduced in the "num_masks" flag, which serves as the basis for identifying the food to be segmented using FoodSAM's semantic segmenter, SETR. The semantic segmenter operates as follows:

1. The model configuration is loaded from a file.

2. "cudnn_benchmark" optimization is enabled for convolution operations on GPUs if specified in the configuration.

3. Pre-trained model usage is disabled, and the test mode is set.

4. The segmentation model is initialized with the provided configuration and checkpoint. A data transformation pipeline using the model configuration is built for the test phase. Input data is prepared for inference, i.e., the transformations defined in the test pipeline are applied to the input images. The data is distributed to the specified GPU.

5. Semantic segmentation prediction is performed on the specified images.

6. The result (the masks with the semantic segmentation of the food in the images) is saved with the specified visualization options at the specified path. Examples of the outputs for num_masks = 1 and num_masks = 3 are shown in Fig. 9 (a) and (b), respectively.

7. Optional: The overlay is saved at the specified path. An overlay example is shown in Fig. 9 (c).

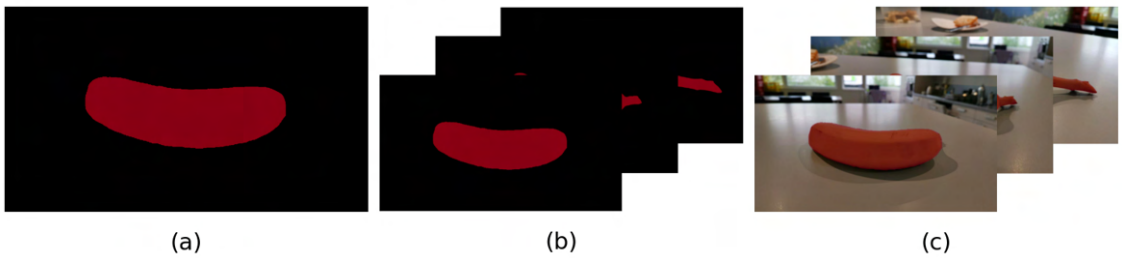


Figure 9: Outputs from our framework steps. (a) shows the SETR semantic segmentation on the first frame, followed by (b), which shows the SETR semantic segmentation for num_masks = 3, and (c), which shows the overlays of the generated masks on their respective frames.

Notably, the model is pre-trained (i.e. transfer learning), and we have the checkpoint, so there is no need to train the model in each execution. SETR [10] was used as the baseline for semantic segmentation. The encoder and the decoder were ViT-16/B and MLA, respectively. The semantic segmentation model is a deep network model that uses a variant of the ViT called Multi-Level Aggregation (MLA) in an Encoder-Decoder approach. Model components and configurations are summarized as follows:

1. **Backbone:** It is a ViT_MLA model with a base ViT architecture, patch size 16, embedding dimensions 768, 3 input channels, 12 layers, 12 attention heads, and 19 output classes.

2. **Decode head:** It is a ViT_MLAHead model with 1024 input channels from the encoder output and MLA bottleneck, 512 channels, 104 output classes, and utilizes cross-entropy loss without a sigmoid function.

3. **Auxiliary head:** It lists 4 auxiliary heads for semantic segmentation, with configurations similar to decode head but with 256 input channels and cross-entropy loss with sigmoid function as loss configuration.

Additionally, an SGD-based optimizer [26] with a learning rate of 0.002 is used, and the learning rate adjustment policy is polynomial [27]. As said previously, we employ the SETR model to generate segment masks for food items. Comprising 251 layers, the SETR architecture initiates with Conv2d, PatchEmbed, and Dropout layers. It then iterates through a sequence of LayerNorm, Linear, Dropout, Linear, Dropout, Attention, Identity, LayerNorm, Linear, GELU, Dropout, Linear, Dropout, Mlp, Identity, and Block layers until reaching layer 196, where three LayerNorm layers are introduced. Another cycle follows, featuring Conv2d, SyncBatchNorm, and ReLU layers. At layer 224, Conv_MLA and VIT_MLA emerge, while layer 226 reintroduces the sequence of Conv2d, SyncBatchNorm, and ReLU layers. Finally, at layer 250, MLAHead and Conv2d layers are incorporated. The model boasts a total parameter count of 91051880, with an input shape of (3, 768, 768) and an output shape of (-1, 104, 192, 192).

3.4.2.2 XMem++

Once the SETR part has been executed, the next step is to apply the detected food segmentation to the rest of the video using XMem++. SETR (see Sec. 3.4.2.1) saves the masks obtained by its model to the specified path. Therefore, XMem++ reads these masks and executes its algorithm based on them. The model first runs the inference function on the video, processing each frame to generate semantic segmentation masks. Here is a description of how the algorithm works:

1. The GPU is set as the computing device, and gradient calculation is turned off since XMem++ comes pre-trained.

2. The main objects necessary for video processing, such as label mappers, sample processors, video readers, and data loaders, are loaded. The segmentation model is loaded.

3. The frames that need to be loaded into permanent memory are determined based on the presence of associated masks. Typically, only the first frame will be stored. It retrieves the frame and its corresponding mask, converts the mask using the mapper, and adds the frame-mask pair to the permanent memory of the inference processor.

4. Each frame of the video is processed one by one. Inference is performed on the frame using the semantic segmentation model. The inferred masks are adjusted as necessary, and the inferred mask is saved. The first mask is ignored because it is already in permanent memory. For each frame, it is checked whether segmentation in the image is necessary. This involves encoding a key from the image, calculating features at different scales, and

using memory to compare and update the segmentation if necessary. If an input mask is provided, it updates the predicted segmentation. Finally, the processed segmentation of the image is returned.

5. The inferred masks are saved to disk for further use.

6. Optional: The overlays are saved to disk.

3.4.3 Output

We explain the outputs that are obtained during the workflow, the cases are shown visually in Fig. 10:

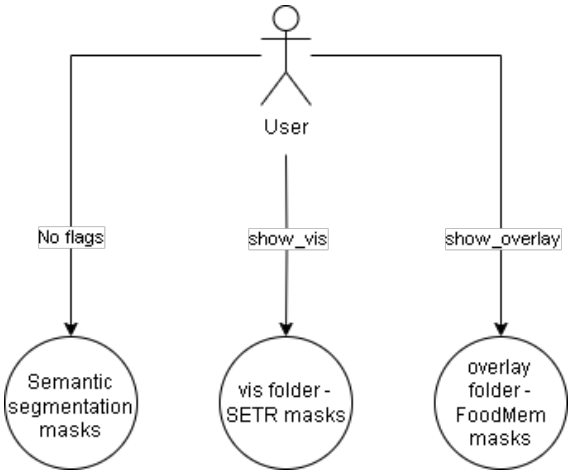


Figure 10: Use-case diagram of the outputs depending on the visualization flags the user introduces. If no flag is entered, semantic segmentation masks will be obtained; the user can create the "vis" folder if the show_vis flag is entered; the user can create the "overlay" folder if the show_overlay is entered.

- If executed without any flags, semantic segmentation masks will be obtained for each frame, saved in the path specified in `"--output"`.
- If the `"--show_vis"` flag is executed, a directory named "vis" will be additionally generated within the path specified in `"--output"`, where the overlay of the masks generated by SETR [10] on the original images will be stored.
- If the `"--show_overlay"` flag is executed, a directory named "overlay" will be additionally generated within the path specified in `"--output"`, where the overlay of the masks on their respective original images will be stored.

3.5 Our dataset

Here, we explain our dataset to validate the proper functioning of our framework. We used two datasets: Nutrition5k [11] and Vegetables & Fruits (V&F) [12]. Both

datasets include images and extra information such as IMU data. Upon examining and researching these datasets, we realized that they are large (e.g. 5000 dishes in Nutrition 5k), covering many dishes with multiple frames for each dish. We selected a subset of dishes from each dataset to simplify the work and ensure efficiency in the validation process. Our selection is based on the variation of the food, the complexity of the video, and the number of ingredients per video, trying to have excessively simple and, at the same time, extremely complex scenes within the dataset.

To solve the problem of duplicate frames, we used Imagededup [28], a Python library, to find and remove duplicate images from datasets. For the correct project evaluation, we needed the ground truth semantic segmentation (see Sec. 5.2). We manually made the ground truth for each frame of every dish selected since it was not provided by the datasets. We used LabelMe [29], an open-source annotation used for labelling and annotating images.

The food category labels annotated by LabelMe were based on the official food classification from the United States Department of Agriculture (USDA). The classification shown in Fig. 11 provides a coherent and logical framework for categorizing a wide range of foods into meaningful groups. Using this classification, we ensured that our food categories were according to standards already recognized in the food industry and scientific research.

Food Group	Subgroups	Sample Foods
Fruits	Whole Fruit* Fruit Juice*	apple, banana, orange, peach, pear, grapes, watermelon, cantaloupe, pomegranate, strawberry, pineapple, mango, raisins, grapefruit, cherries, raisins, 100% fruit juice, etc.
Vegetables	Dark Green Vegetables	broccoli, collard greens, spinach, romaine, etc.
	Red and Orange Vegetables	carrots, red peppers, tomatoes, sweet potatoes, etc.
	Beans and Peas**	kidney beans, black beans, chickpeas, split peas, lentils, etc.
	Starchy Vegetables	white potatoes, corn, green peas, etc.
	Other Vegetables	mushrooms, summer squash, iceberg lettuce, avocado, etc.
Grains	Whole Grains	whole wheat bread, brown rice, popcorn, oatmeal, etc.
	Refined Grains	pretzels, English muffins, corn tortilla, grits, regular pasta, etc.
Protein Foods	Seafood	salmon, tuna, trout, tilapia, sardines, herring, mackerel, shrimp, crab, oysters, mussels, etc.
	Meat, Poultry, and Eggs	beef, chicken, turkey, pork, eggs, etc.
	Nuts, Seeds, and Soy	nuts, nut butters, seeds, soy products, etc.
Dairy	Milk and Yogurt* Cheese*	milk, yogurt, kefir, cheese, cottage cheese, calcium-fortified soymilk, etc.
*While the Fruits Group and Dairy Group do not technically have subgroups, the *2015-2020 Dietary Guidelines for Americans* recommend that Americans consume more whole fruits than fruit juice and more milk and yogurt than cheese. Whole fruit includes fresh, canned, frozen, dried, cut-up, and pureed fruit.		
**Beans and peas can be considered part of the Vegetables Group or Protein Foods Group. They include key nutrients from both food groups.		
Note: Americans are encouraged to choose foods in their most nutrient dense forms and to drink and eat less sodium, saturated fat, and added sugars. For example, in the Protein Foods group beef, chicken, turkey, and pork should be lean or skinless. Choices from the Dairy group should be low-fat or fat-free. And for canned vegetables, choose low sodium or no salt added options.		

Figure 11: USDA food classification table, from [30]. This table categorizes various foods into five main groups: Fruits, Vegetables, Grains, Protein Foods, and Dairy. Each group is subdivided into specific subgroups with representative sample foods.

Our dataset includes 31 dishes from Nutrition5k with 1356 annotated frames and 11 dishes from V&F with 2308 annotated frames, for a total of 42 dishes with 3664 annotated frames. By choosing a subset of images and ensuring annotation accuracy

in each frame, we created a robust dataset for effectively testing and validating our framework.

4 FoodMem Implementation

In this section, we discuss the range of technologies, tools and software engineering techniques used during the development and execution of our framework. Each plays an important role in different steps of the project lifecycle. The following subsections provide detailed views on the system environment, development tools, programming languages, version control systems and agile methodologies used in this project. Understanding these technologies' capabilities and application areas will provide insight into how they work collectively to ensure the successful realization of our goals.

4.1 Windows

Windows is the operating system used to develop and test our framework. Due to its user-friendly interface and extensive compatibility with various software and development tools, it serves as the primary platform for building our project. Key features of Windows that are beneficial for our framework include:

- **Wide range of development tools:** Windows supports a wide range of integrated development environments (IDEs) and text editors, including IntelliJ IDEA, which was used for developing our framework.
- **Ease of use:** The graphical user interface (GUI) of Windows makes it easy to navigate and manage files and applications, enhancing productivity and reducing the learning curve.
- **Compatibility with Docker:** Docker, which was used for containerization, can be easily installed and run on Windows. This facilitated the evaluation comparison between our framework and other models with Linux-specific instructions.
- **Python support:** Python, the primary programming language used in our framework, runs smoothly on Windows. This allowed for the execution of scripts, management of dependencies, and development of functionalities without compatibility issues.
- **Integrated CLIs:** Windows provides powerful command-line tools, such as PowerShell and Command Prompt, which were useful for running scripts, managing version control with Git, and performing various administrative tasks.

4.2 IntelliJ IDEA

IntelliJ IDEA is an integrated development environment extensively used to develop our framework. It is known for its powerful features and extensive support for various programming languages and frameworks. IntelliJ IDEA provided the following advantages for our project:

- **Comprehensive development:** IntelliJ IDEA offers a wide range of built-in tools and features, such as code analysis, refactoring, and debugging, which streamline the development process and improve code quality.
- **Python support:** IntelliJ IDEA has excellent support for Python through the Python plugin. This allowed us to write, test, and debug Python code efficiently, which is the primary programming language used in our framework.
- **Version control integration:** The IDE provides seamless integration with Git and GitHub, enabling easy version control. We could commit changes, merge branches, and resolve conflicts directly within the IDE.
- **User-friendly interface:** IntelliJ IDEA's intuitive and customizable interface enhanced productivity by allowing us to arrange tools and windows according to our preferences. This made navigation and project management more efficient.
- **Code completion and suggestions:** The intelligent code completion and suggestions feature helped speed development by predicting the next lines of code and providing useful hints, reducing the likelihood of errors.
- **Plugin ecosystem:** IntelliJ IDEA has a rich ecosystem of plugins that can extend its functionality. We utilized various plugins to enhance our development environment, including those for Docker and Python.

4.3 Python

Python was the primary programming language used in the development of our framework. Its versatility, extensive libraries, and ease of use made it an ideal choice for our project. Here are the key reasons why Python was integral to our framework:

- **Ease of use and readability:** Python's simple syntax and readability allowed for rapid development and easy codebase maintenance. This was particularly important for a complex project like FoodMem, where clarity and ease of understanding are important.
- **Extensive libraries and frameworks:** Python has a lot of libraries and frameworks that facilitated various aspects of the project. For instance, we used OpenCV [31] for image processing, NumPy [32] for numerical computations, and PyTorch [33] for implementing and training machine learning models. These libraries significantly reduced development time and effort.
- **Strong support of machine learning:** Python is a popular language in the machine learning community, with strong support for machine learning and deep learning frameworks such as PyTorch. This allowed us to leverage state-of-the-art algorithms and models for the semantic segmentation tasks in our framework.

- **Community and documentation:** Python's large and active community provided extensive documentation, tutorials, and forums that were invaluable for troubleshooting and finding solutions to challenges encountered during development.

4.4 Conda

Conda is a powerful package and environment management system widely used in data science and software development projects. It efficiently manages dependencies, libraries, and software environments, ensuring the project's environment remains consistent across different development and deployment stages. Conda played an important role in the development of our framework and in maintaining a stable and reproducible environment. Here are the key elements of Conda used in our framework project:

- **Environment management:** Conda allowed us to create isolated environments for the project, ensuring that all dependencies and packages required by our framework were contained within a specific environment. This isolation prevented conflicts between different package versions and made managing and updating dependencies easier.
- **Dependency management:** With Conda, we could specify and install all the necessary libraries and tools required for our framework. This included essential packages for machine learning, image processing, and other scientific computing needs, such as PyTorch, OpenCV, NumPy, and more.
- **Portability:** By using Conda environment files (YAML files), we could easily share the exact environment setup with other team members and ensure that anyone had the same setup. This portability is vital for projects, minimizing issues related to environmental discrepancies.
- **Cross-platform compatibility:** Conda supports multiple operating systems, including Windows, macOS, and Linux. This compatibility was beneficial as it allowed team members to work on the project across different platforms without encountering dependency issues.

4.5 LabelMe

LabelMe is an open-source annotation tool widely used to create labelled datasets for computer vision applications. It allows users to manually annotate images with polygons, rectangles, circles, lines, and points. In the FoodMem project, LabelMe played an important role in generating the ground truth data needed to evaluate the performance of the segmentation models. Here is a detailed explanation of its functionality and how it was used:

- **Annotation tool:** LabelMe provides an intuitive interface for manually labelling objects in images. Users can draw various shapes around objects to create detailed annotations.
- **Open-source and extensible:** As an open-source tool, LabelMe can be customized and extended to suit specific project needs, making it versatile for various annotation tasks.
- **Managing large datasets:** Given the large datasets, LabelMe's user-friendly interface and efficient annotation capabilities were essential for effectively handling and labelling the images.
- **Attribute tagging:** Annotators can add attributes to the annotations, such as labels or categories, which are useful for detailed segmentation tasks.
- **User-friendly:** Its intuitive interface makes it accessible even for those who are not experts in annotation tasks.
- **Community support:** Being open-source, LabelMe has a large community of users and developers, offering support and contributing to its continuous improvement.
- **Ground truth creation:** For evaluating our framework, accurate ground truth segmentation masks were necessary. Since the datasets Nutrition5k and Vegetables & Fruits lacked these masks, we used LabelMe to annotate each frame manually.
- **Annotation process:** The process involved manually drawing polygons around food items in each frame. Although time-consuming, this step was important for obtaining high-quality, accurate annotations.

4.6 Imagededup

Imagededup is a Python library designed to detect duplicate images in a dataset. It uses various techniques, such as perceptual hashing, to identify visually similar images, even with different resolutions or slight modifications. In our pipeline, Imagededup was used to ensure the dataset's quality by removing redundant video frames. This step was important to enhance the accuracy and reliability of the segmentation model. FoodMem could keep a clean and diverse set of images, leading to better model performance and more precise food segmentation results.

4.7 Mozaic

Mozaic is a tool designed to streamline the creation of compound figures for visual comparisons. It automates the process of combining multiple images into a single figure, minimizing the need for manual editing. In our framework, Mozaic was used to generate segmentation result images and facilitated visualization by easily

comparing different frames and segmentation methods. Using Mozaic, we could create detailed and informative compound figures, enhancing our qualitative results' presentation. Mozaic was developed by the GCVCG team at the University of Barcelona and can be found on GCVCG's GitHub page [24].

4.8 Bash

Bash [34], which stands for "Bourne Again Shell," is a command language interpreter typically used in Unix-based operating systems. For our framework, Bash was utilized for the following purposes:

- **Script execution:** Bash scripts were created to automate various tasks related to project setup.
- **File and directory management:** Bash commands were used to navigate the project directory structure and manipulate files.

4.9 Docker

Docker [35] is a platform for developers to package, distribute, and run applications within lightweight, portable containers. These containers encapsulate all the necessary dependencies and configurations required to run an application, ensuring consistent behavior across different computing environments.

We used Docker to address compatibility issues between DEVA and the Windows operating system. To overcome this obstacle and facilitate comparison experiments between DEVA and our framework, Docker was used as a solution. By encapsulating DEVA and its dependencies within a Docker container, we effectively abstracted away the underlying operating system differences. This approach allowed us to run DEVA seamlessly on Windows systems without requiring manual configuration or resolving dependency conflicts. Docker worked as a bridge between the Linux-based DEVA model and the Windows environment, providing a consistent and isolated runtime environment for conducting comparative experiments with our framework. This utilization of Docker underscores its versatility in enabling cross-platform compatibility and facilitating the seamless integration of diverse tools and technologies within the development workflow.

4.10 Git

Git provides a distributed and decentralized platform for tracking changes to project files, enabling efficient collaboration, code review, and version management. In the FoodMem project, Git was utilized as the primary version control system for tracking changes to the source code, scripts, and documentation. By using Git, we could modify code, experiment with new features, and address issues while maintaining a coherent history of changes. Moreover, Git's branching and merging

capabilities facilitated the implementation of various development workflows, such as feature branching, bug fixing, and release management. Separate branches were created to work on specific tasks or features, such as isolating experimental changes, and later merge them back into the main codebase after review and testing.

4.11 GitHub

GitHub is the hosting platform for the FoodMem project's code repository. GitHub is a central hub where project code, documentation, and related resources are stored, managed, and shared. GitHub provides several key functionalities and benefits:

- **Code hosting:** GitHub hosts the project's Git repository, allowing team members to push, pull, and clone code from a centralized location. This enables seamless collaboration and version control, ensuring all team members can access the latest project codebase.
- **Collaboration tools:** GitHub offers collaboration features such as issue tracking, pull requests, and code reviews. Team members can create and assign issues, discuss project tasks, and propose changes via pull requests. This promotes transparent communication and effective coordination among team members.
- **Documentation management:** GitHub's support for markdown formatting allows for creating detailed project documentation directly within the repository. README files, wikis, and documentation pages can be used to provide project overviews, installation instructions, usage guidelines, and contribution guidelines.

4.12 Communications mediums

Effective communication is crucial for the success of our project. We utilized communication tools to ensure seamless coordination and information sharing among team members. The primary communication mediums we used included Slack, a WhatsApp group called FoodMem, Gmail, and Google Meet. Slack was our main platform for daily communication and collaboration. The key features of Slack that benefited our project were channels that allowed us to organize conversation topics, making it easier to find and reference past discussions. File-sharing capabilities enabled us to share quickly and access files, including code snippets, documents, and images. Additionally, Slack's integration with other tools and services, such as GitHub and Google Drive, streamlined our workflow and kept everything interconnected. Real-time communication facilitated instant messaging and quick responses, enhancing our ability to address issues promptly. The WhatsApp group named FoodMem served as a more informal and immediate communication channel. It was beneficial for sharing quick updates or urgent messages when team members were away from their computers. Coordination of meetings and discussing schedules more conversationally was also a key benefit. This platform allowed us to stay

connected and responsive. Gmail was used for more formal communications and documentation purposes. It played a significant role in sending and receiving official communications, such as meeting invitations, and progress reports. Document sharing was facilitated through email attachments, allowing us to distribute larger documents and files. Additionally, Gmail provided an efficient way to archive important communications and documents for future reference. This medium ensured that all critical information was documented and accessible, contributing to the overall organization of the project. Google Meet was our go-to platform for virtual meetings. It was crucial in facilitating face-to-face discussions, essential for doubts resolution, important announcements, and regular check-ins. The screen-sharing feature allowed team members to share their screens, which was particularly useful for demonstrating code and reviewing documents. This platform ensured that our virtual interactions were productive and effective.

4.13 Agile methodologies

The FoodMem project used various software engineering techniques to ensure effective project management and development practices. Agile methodologies, specifically SCRUMBAN, were applied to our framework to simplify the workflow.

Agile methodologies [36] were adopted to provide a flexible and iterative approach to software development. This approach emphasizes continuous feedback, collaboration, and small, incremental changes rather than a monotonous development process. Agile methodologies helped to change and respond to new requirements and challenges quickly.

4.13.1 SCRUMBAN

SCRUMBAN is a hybrid project management framework that combines the structured approach of SCRUM with the visual workflow management of KANBAN. This integration leverages the strengths of both methodologies to optimize workflow, enhance team collaboration, and ensure efficient project delivery. SCRUMBAN provides a flexible yet structured environment that adapts to changing project requirements while maintaining a continuous rate of development.

4.13.1.1 SCRUM

SCRUM [37] is an agile framework that structures the development process into fixed-length iterations called sprints. It focuses on iterative progress, transparency, and collaboration. Key elements of SCRUM used in the FoodMem project include:

- **Sprints:** The development process is divided into sprints, each lasting two weeks. These sprints focus on completing a predefined set of tasks from the product backlog.

- **Daily stand-ups:** Short, daily meetings help the team discuss progress, plan the day's work, and address any impediments. These meetings ensure transparency and keep the team aligned. We used Google Meet for these stand-ups.
- **Sprint planning:** At the beginning of each sprint, the team plans which tasks to complete, setting clear goals and defining the sprint backlog.
- **Sprint review and retrospective:** At the end of each sprint, the supervisors review the completed work and hold a retrospective to discuss what went well, what could be improved, and how to implement those improvements in future sprints.
- **Global meetings:** We held a global meeting with supervisors weekly to review progress and align on the next steps. These meetings were conducted in person, at university, except when any team member had an impediment, in which case we used Google Meet.

4.13.1.2 KANBAN

KANBAN [38] is a visual workflow management method that helps teams manage tasks and optimize processes. It focuses on continuous delivery and efficiency. Key elements of KANBAN used in the FoodMem project include:

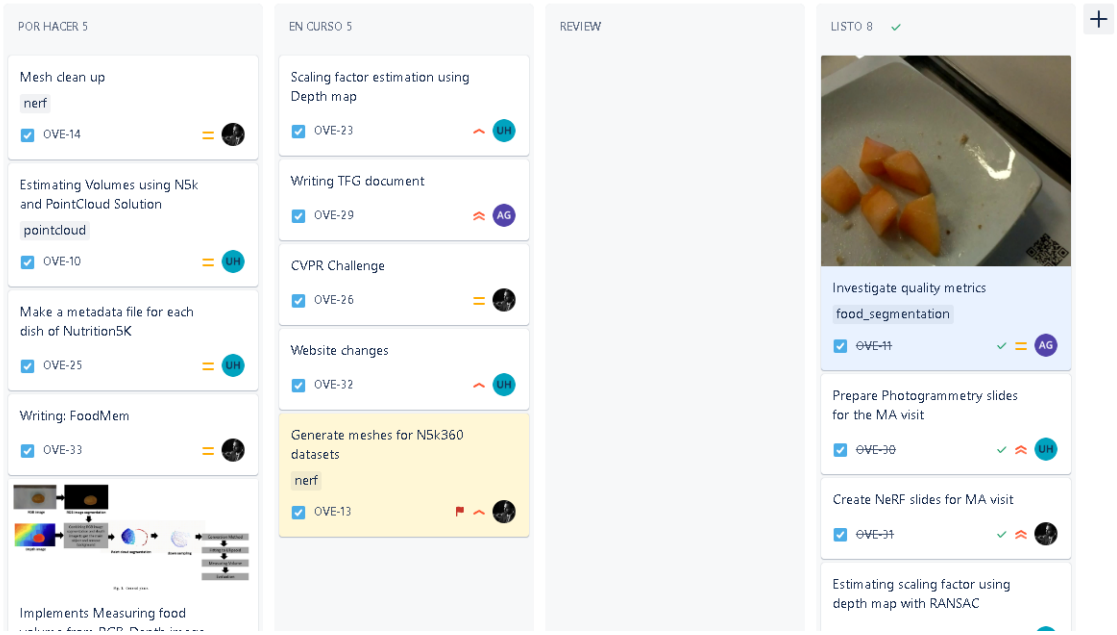


Figure 12: VolE board. This is a portion of the KANBAN board we used to organize the project. It includes tasks from the TFG and VolE project.

- **Visual management:** A KANBAN board provides a visual representation of tasks and their status, divided into columns such as "To Do", "In progress",

"Review" and "Done". This promotes transparency and accountability. We used JIRA to manage our KANBAN board, as shown in Fig. 12.

- **Continuous delivery:** KANBAN facilitates a continuous flow of work, enabling the team to deliver small, manageable pieces of functionality regularly. This ensures continuous progress and quick adaptation to changes.
- **Documentation:** We used Confluence alongside JIRA to document our progress, decisions, difficulties, and plans, ensuring all team members had access to up-to-date project information.

By combining SCRUM's structured approach with KANBAN's flexibility and visual management, SCRUMBAN offers a balanced framework that enhances workflow efficiency, improves team collaboration, and ensures the timely delivery of high-quality software. The FoodMem project benefited from this hybrid approach by maintaining a structured yet adaptable development process, meeting deadlines, and effectively managing project requirements.

5 Validation

We assess our framework performance through an evaluation process. First, we explain the used datasets and the preprocessing pipeline. Then, the evaluation process covers quality metrics, result analysis, and comparison to state-of-the-art methods. Additionally, we outline the implementation settings for transparency and reproducibility. Plus, we realized an ablation study to examine the effects of modifying the masks produced by SETR. Finally, we outline the limitations of our framework, providing context for the findings and guiding further research.

5.1 Datasets

Two principal datasets used in developing our framework were Nutrition5k and Vegetables & Fruits. These datasets were used to validate segmentation models, which assure accurate and effective food recognition and segmentation.

5.1.1 Nutrition5k

The Nutrition5k dataset is a comprehensive collection of images that predominantly feature various types of foods. It is a crucial dataset for researchers and scientists involved in food recognition and segmentation tasks. This dataset primarily consists of bounded scenes, where the camera follows a predefined path or trajectory to capture images systematically. This controlled setup ensures consistent image capture conditions, making it ideal for developing and testing food recognition algorithms. These bounded scenes minimize the variability and unpredictability that often arise in unbounded scenes, thereby enhancing the reliability and accuracy of segmentation tasks (see Sec. 1.3). Some key features of the Nutrition5k dataset include:

1. **Imagery:**

    - **Realsense overhead:** This includes depth color images, depth raw images, camera location, and RGB images, offering multiple perspectives and data types for each dish. The camera used to collect the overhead data is an Intel RealSense D435.
    - **Side angles:** This subset contains images of 4799 dishes, each captured in 130 to 560 frames, providing extensive visual information from different angles. The images are collected from five cameras oriented above and around the plate, with one pointing down directly overhead and the other four from each side of the dishes. The four side-angle cameras sweep 90° simultaneously, capturing the full 360°. Fig. 13 visually represents the camera box used in Nutrition5k.

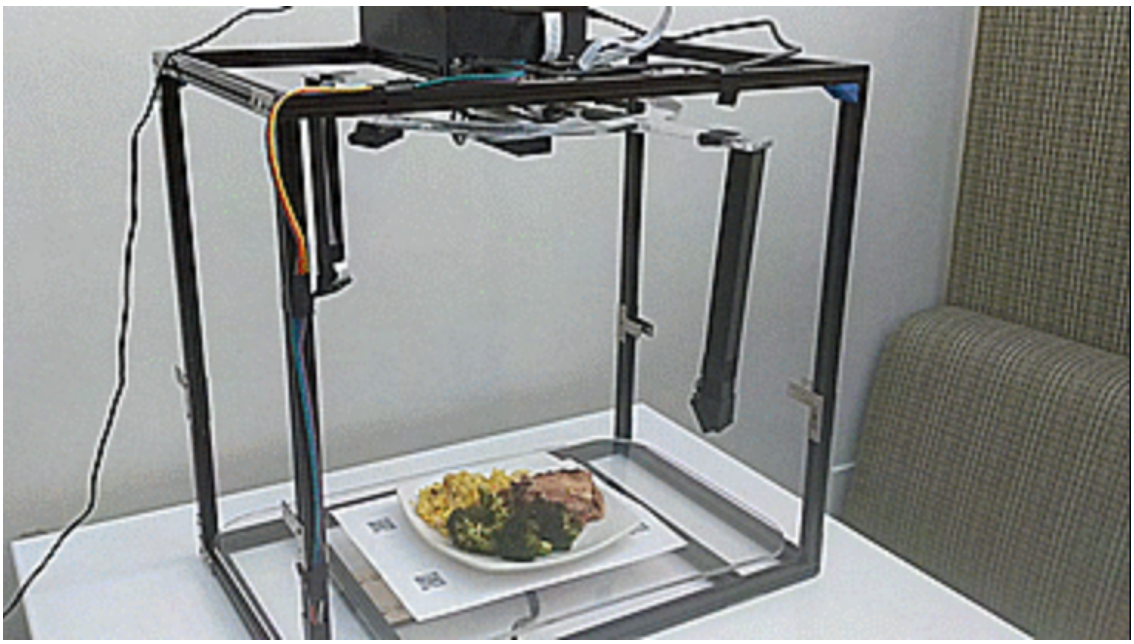


Figure 13: Image of the camera box used in Nutrition5k, from [11]. The camera box is utilized to capture the 360° of the plate. The box has four cameras from each side of the dish and one above the dish. Note that the cameras follow a predefined path capturing images.

2. **Metadata:**

   - **Dish metadata:** This file contains metadata related to each dish, providing contextual information, such as dish ID and ingredient names with their locations in the dish.
   - **Ingredients metadata:** This file includes detailed information about each ingredient, such as ingredient name, ID, calories per gram, fat, carbohydrate, and protein content.

Nutrition5k (see Sec. 3.5) dataset contains about 5000 dishes, a huge collection that brings diversity to the research and analysis of food items. For complete validation of our framework, we choose 31 plates. This selection strategy is intended to include dishes with varied compositions of ingredients for validation of our segmentation model. Each plate has been carefully selected and examined for a diverse ingredient range, increasing the scope of our framework validation. We tested the segmentation model across various food compositions by including plates with distinct ingredient combinations. This broad scope ensured testing our framework's segmentation performance across many categories of meals.

Each plate in the selected subset of the Nutrition5k dataset has several hundred frames; specifically, there are between 130 and 560 frames per plate in this dataset. To simplify the processing and reduce the computational load, Imagededup [28] used an image processing pipeline to compress these frames while preserving the visual information required for segmentation analysis.

Through the pipeline, the number of frames per plate was reduced to about 20 to 65 frames. This made it possible to reduce the computational resources required in the pipeline but ensured that there was enough representation of the main dish's visual characteristics. The keyframes were selected strategically to capture significant visual changes or transitions within each plate. This ensured that critical visual information for segmentation analysis was still contained, which is very important for model evaluation to be done accurately and efficiently. The pipeline process is detailed in Sec. 5.2. With the application of this image processing pipeline, the Nutrition5k dataset is ready for robust validation of our framework segmentation capabilities on diverse culinary compositions.

5.1.2 Vegetables & Fruits

The Vegetables & Fruits dataset consists of various fruits and vegetables like apples, avocados, bananas, blackberries, blueberries, carrots, cucumbers, grapes, peaches, pears, and strawberries. Each food category in the dataset contains between 10 and 15 different scenes, making the last scene of each food item the most complex. Within each scene is a file showing the locations of the fruits and vegetables. This dataset primarily features unbounded scenes, where the camera is not restricted to a predefined path and can move freely. This setup introduces variability in camera angles, distances, and motion dynamics, which reflects more natural, real-world conditions and poses greater challenges for segmentation and tracking tasks (see Sec. 1.3). In addition, there is a directory including images. Normally, these scenes vary from 100 to 600 frames, providing a huge amount of visual data to be analyzed.

We selected each food item's final scenes for our dataset creation process to challenge our framework with complex scenes. We also used Imagededup to reduce the number of frames to a reasonable number and control the overlapping between frames, as mentioned in Sec. 5.1.1, without losing the difficulty of the images needed to evaluate our framework. The pipeline process is detailed in Sec. 5.2.

5.2 Image near similarity

Imagededup is a Python library developed for finding and removing duplicate images from datasets. It provides functionalities for detecting and deleting duplicate and near-duplicate images based on visual similarity. The library uses advanced algorithms to compare the contents of images and detect similarities, therefore facilitating the cleaning of redundant images. These are the algorithms involved in image deduplication:

- **Convolutional neural network (CNN):** This deep learning model can learn and extract features from images. Imagededup provides prepackaged CNN models that can be used for image deduplication tasks. Users can also integrate custom CNN models tailored to their specific needs.
- **Perceptual hashing (PHash):** PHash generates compact hash codes for images based on their visual content. These hash codes capture perceptual

similarities between images, enabling the detection of near-duplicate images with slight variations.

- **Difference hashing (DHash):** DHash computes hash values by calculating the pixel-wise differences between adjacent image pixels. It is effective in identifying images with minor variations or distortions.
- **Wavelet hashing (WHash):** WHash utilizes wavelet transforms [39] to generate hash codes for images. This technique decomposes images into different frequency bands, allowing for the capture of both high-frequency and low-frequency components in image representations.
- **Average hashing (AHash):** AHash creates hash codes by computing the average pixel intensity values of grayscale images. It provides a simple yet effective method for detecting duplicate images based on their overall visual similarity.

We decided that PHash is the best algorithm to preprocess our dataset due to the following factors:

- **Robustness to common image transformations:** PHash creates a compact hash representation of an image on its perceptual features. This hash is peculiarly invariant to all common image transformations, such as scaling, rotation, and small changes in brightness or contrast. Therefore, images with visual similarity, but after some minor changes, can still lead to similar hash values and make PHash good for near-duplicate image detection.
- **Computational efficiency:** Most PHash algorithms are basically fast and, therefore, applicable in image deduplication tasks on a large scale. The perceptible features of the image are extracted and encoded into a fixed-length hash code, which enables the comparison between the hashes of other images using simple distance methods.
- **Compact representation:** For every image, PHash generates a small binary hash code, normally consisting of a few bits. This compact form allows for efficient storage and comparison of image fingerprints, making it practical for indexing and searching large image datasets.
- **Low sensitivity to noise:** PHash algorithms have been implemented to provide robustness to noise and slight variations in the input image. They deal more with high-level perceptual features than with precise pixel values, which ensures that image compression artefacts and sensor noise, among other distortions, have minimal impact.

In general, effectiveness, efficiency, and robustness make PHash a favorite image deduplication and similarity detection tool in many applications. However, the performance of the PHash algorithms needs to be tested on concrete cases and datasets to meet the requirements in terms of accuracy and scalability.

5.2.1 Maximum hamming distance threshold

The maximum hamming distance threshold [40] refers to a parameter of image deduplication algorithms and, more precisely, to the ones based on perceptual hashing, like PHash. In these algorithms, images are transformed into compact hash codes that describe the image's visual content compactly. The hamming distance is the metric for comparing hash codes to identify duplicate and near-duplicate images, which calculates how similar two hash codes are. The hamming distance threshold sets up the maximum allowed difference for two hash codes so that the images are still considered duplicates or near-duplicates. By setting an appropriate threshold for the maximum hamming distance, users can manage the level of sensitivity desired in the deduplication process. A lower threshold will yield a much stricter criterion for declaring duplicates, thereby possibly reducing the chance of false positives but also increasing the chance of missing similar images. A higher threshold may detect more matches, including images with slight variations, but it may also increase the risk of false positives. The optimum maximum hamming distance threshold is normally a trade-off between precision and recall, dependent on the dataset's specific requirements and characteristics. Experimentation and fine-tuning may be required to find the most suitable threshold for a given application.

We did a comparative evaluation process using different distance threshold values to find the best value. We created Fig. 14 to show how the number of frames changes by applying the different algorithms that provide Imagededup at different distance thresholds for various dishes and initial frames.

Fig. 14 allows us to compare how the number of frames changes for every dish depending on the maximum distance threshold in the process of image deduplication. From the analysis of these results, we can determine which threshold best balances removing duplicate images and retains relevant information in the remaining frames. This helps us determine the best value for the maximum hamming distance threshold that maximizes the effectiveness of Imagededup for our particular application context. After a comparative evaluation, we found that the maximum hamming distance threshold of 15 gave the best results. This threshold value is balanced, eliminating duplicating images and preserving the necessary information in the remaining frames. Therefore, based on our analysis, the maximum hamming distance threshold of 15 is the most suitable choice for Imagededup's performance in this dataset.

5.3 Quality metrics

We outline the quality metrics for evaluating our framework's performance and effectiveness. Quality metrics are very important for determining the model's accuracy, efficiency, and reliability. Mean Average Precision [41] and recall [42] provide an evaluation of the FoodMem model's performance. We focus on these metrics so that our model is accurate and reliable regarding practical applications for food segmentation tasks. The metrics help us understand the model's ability to detect and segment food items effectively, which would be useful in such applications as

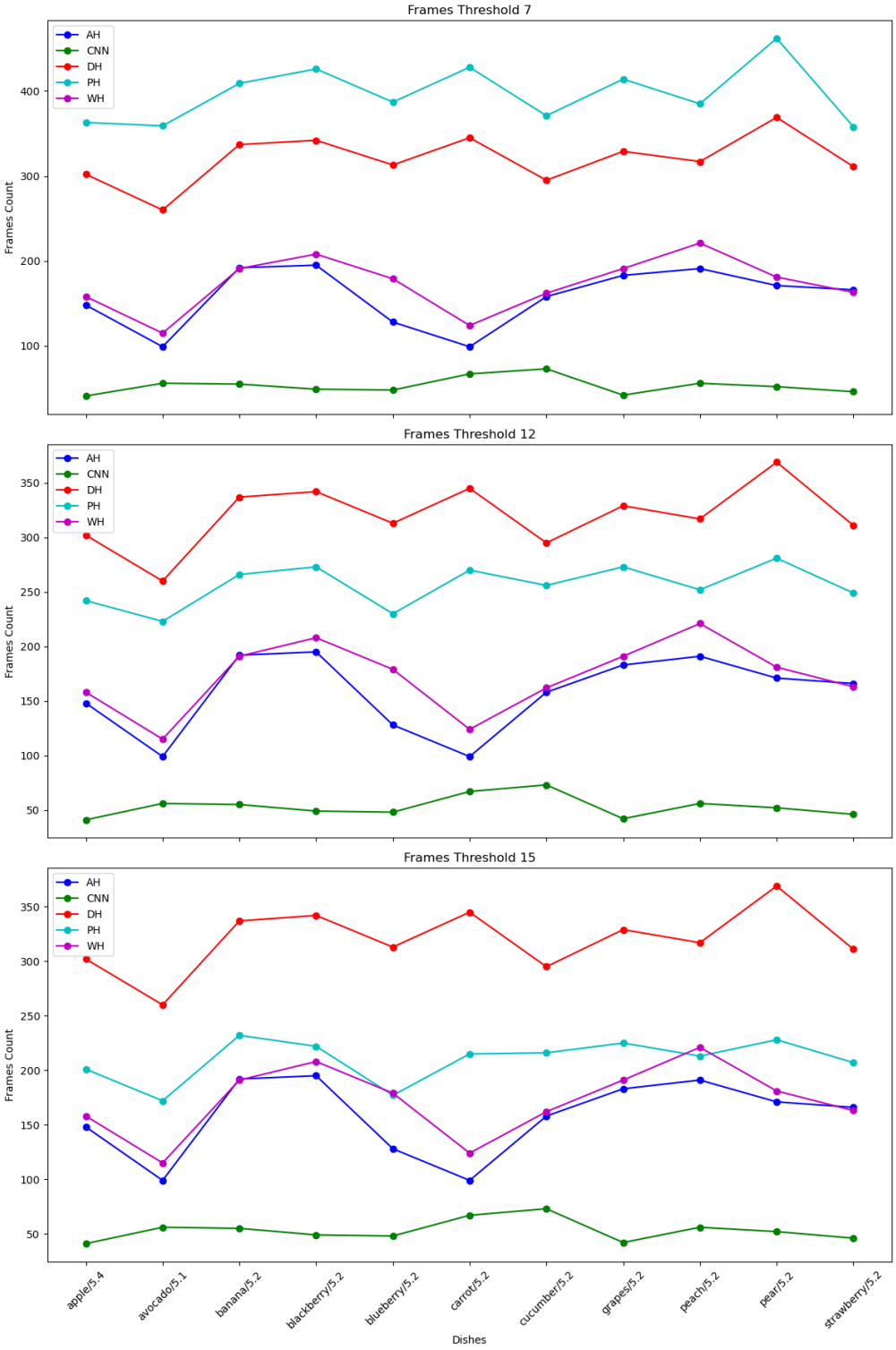


Figure 14: Imagededup methods thresholds comparison. The figure consists of three subfigures, each corresponding to 7, 12 and 15 thresholds respectively. The figures show the number of frames of each dish once the Imagededup method is applied with its maximum hamming distance threshold.

dietary assessment and culinary automation. In the subsections below, we focus on the two key metrics that were employed.

5.3.1 Mean Average Precision

Mean average precision (mAP) is a commonly used metric in object detection and segmentation tasks. It provides a single value summarizing the precision-recall curve for each class and then averages these values over all classes. Mean average precision is calculated by first computing each class's Average Precision (AP), which is the area under the precision-recall curve. The mAP is then the mean of the AP values for all classes.

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{5.1}$$

where P_n and R_n are the precision and recall at the n-th threshold. This implementation does not use interpolation, unlike calculating the area under the precision-recall curve with the trapezoidal rule [43], which uses linear interpolation and can be overly optimistic. Mean average precision comprehensively measures the model's precision and recall performance across different classes. A higher mAP indicates that the model performs well in detecting and segmenting objects accurately.

5.3.2 Recall

Recall, also known as sensitivity or true positive rate, measures the ratio of correctly predicted positive observations to all observations in the actual class. Recall is calculated as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

where TP is the True Positive; FN is the False Negative. Recall is important for evaluating the model's ability to capture all relevant instances of the objectives being segmented. High recall ensures the model does not miss significant objects within the images or video frames.

5.4 Baselines

Comparisons with other state-of-the-art models were needed to evaluate our framework's performance and capabilities effectively. The choices of FoodSAM, DEVA, and kMean++ were strategic, based on their different characteristics and relevance to our specific use case in food segmentation. Comparison models have been chosen to reflect their respective strengths in food segmentation, video analysis, and clustering. These comparisons highlight our framework's improvements in terms of segmentation quality, tracking accuracy, and overall performance. Demonstrating these improvements over such methods can help validate our framework's efficacy

and efficiency for real-world applications. A comparison of our framework with FoodSAM will help us to understand if our framework is a worthwhile improvement, offering a faster execution while achieving nearly the same high-quality results as FoodSAM. The following subsections explain the reasons for each model selection for comparison.

5.4.1 FoodSAM

- **Relevance to food segmentation:** FoodSAM is specifically designed for food semantic segmentation, making it a direct competitor and an appropriate benchmark for our framework.
- **High-quality segmentation:** FoodSAM is known for precisely segmenting food items within images. Comparing our framework to FoodSAM allows us to assess if FoodMem maintains or improves segmentation quality.
- **Single image limitation:** FoodSAM is designed for segmenting single images and takes so much time to segment videos since it processes each frame individually. While this ensures accurate segmentation, it is time-consuming.
- **Efficiency in execution:** Comparing our framework to FoodSAM allows us to determine if our framework is more efficient, offering faster execution times while achieving similar or nearly identical segmentation results.

5.4.2 DEVA

- **Advanced video analysis:** DEVA is recognized for its strong performance in video analysis and object-tracking tasks. This makes it an ideal candidate for comparing video segmentation and tracking capabilities.
- **General-purpose use:** DEVA is designed for general use and not specifically for food items. This broader application scope means it can handle a variety of objects and scenes.
- **Focus on food items:** To ensure that DEVA's performance is relevant to food segmentation, we utilized its text prompt feature to specify "food" for all our dishes during evaluation. This approach helps us measure how well DEVA can adapt to our specific focus on food.
- **Handling complex video frames:** DEVA's ability to handle complex frames in video provides a challenging benchmark. By comparing with DEVA, we can demonstrate how well our framework manages segmentation consistency across frames in a video.
- **Linux dependencies:** DEVA's support of Linux-specific dependencies is needed using Docker in our evaluation process, emphasizing our framework's compatibility and efficiency across different environments.

5.4.3 kMean++

- **Clustering and segmentation:** kMean++ is a robust clustering algorithm often used for segmentation tasks due to its efficiency in initializing centroids and improving clustering performance.
- **Baseline comparison:** kMean++ acts as a strong baseline to compare against traditional clustering methods. This comparison helps illustrate the advancements our framework offers over simpler, yet effective, segmentation techniques.
- **Efficiency and speed:** Evaluating our framework against kMean++ allows us to showcase improvements in execution speed and segmentation quality, particularly in large datasets or videos.

5.5 Results and comparison to SOTA

We present and compare the masks generated by FoodSAM, DEVA, kMean++, and our framework as a qualitative result [44]. We also provide an analysis of the execution time for each model and evaluate their performance using mean average precision and recall metrics as a quantitative result [45].

5.5.1 Mask comparison

We display each model's segmentation masks for the same set of frames. This visual comparison helps illustrate the differences in segmentation quality and accuracy between the models. Since our dataset contains around 50 dishes with multiple frames, we will display the most representative results. After comparing all the following subsections, we notice that our framework offers the most accurate and clean segments compared to the other methods. Masks generated by our framework capture the shape and contour of food well, with less inclusion of unwanted areas and more precise delineation of edges. The segments are more consistent and closer to the ground truth.

5.5.1.1 FoodSAM and FoodMem

Figs. 15 and 16 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by FoodSAM. We applied the Mozaic tool to make the visualizations attractive, as explained in Sec. 4.7.

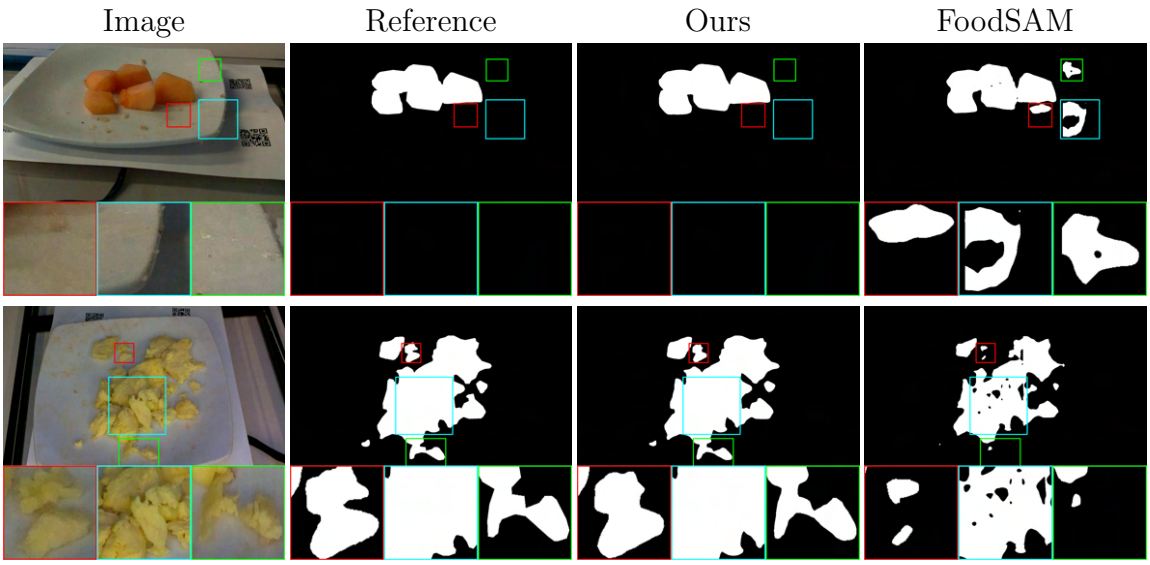


Figure 15: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by FoodSAM. The dataset used is Nutrition5k.

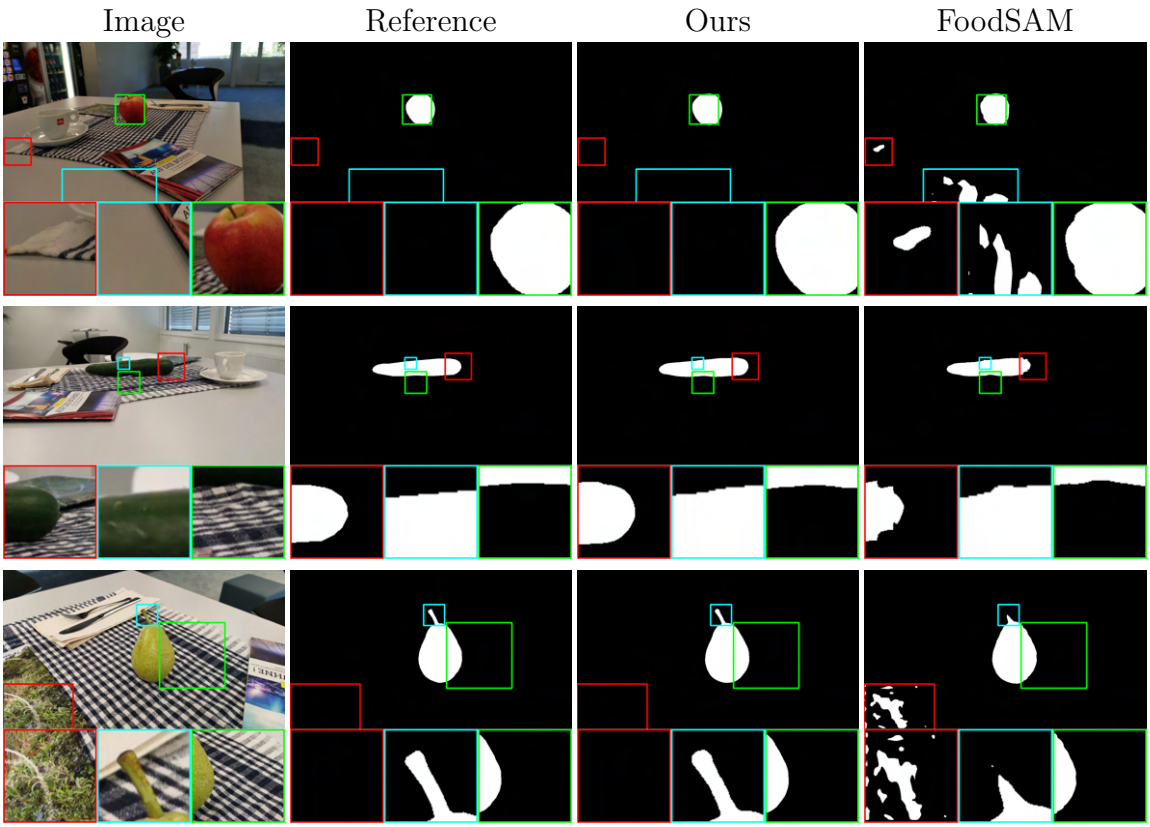


Figure 16: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by FoodSAM. The dataset used is Vegetables & Fruits.

FoodSAM seems to capture the basic shapes and contours of food objects well, but sometimes includes non-relevant areas or fails to capture all the fine details. The masks are consistent compared to other SOTA masks, which are displayed in the following sections, but sometimes lack edge precision and may not include some relevant parts of food.

5.5.1.2 DEVA and FoodMem

Figs. 17 and 18 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by DEVA. We applied Mozaic tool to do the visualizations attractive, as explained in Sec. 4.7.

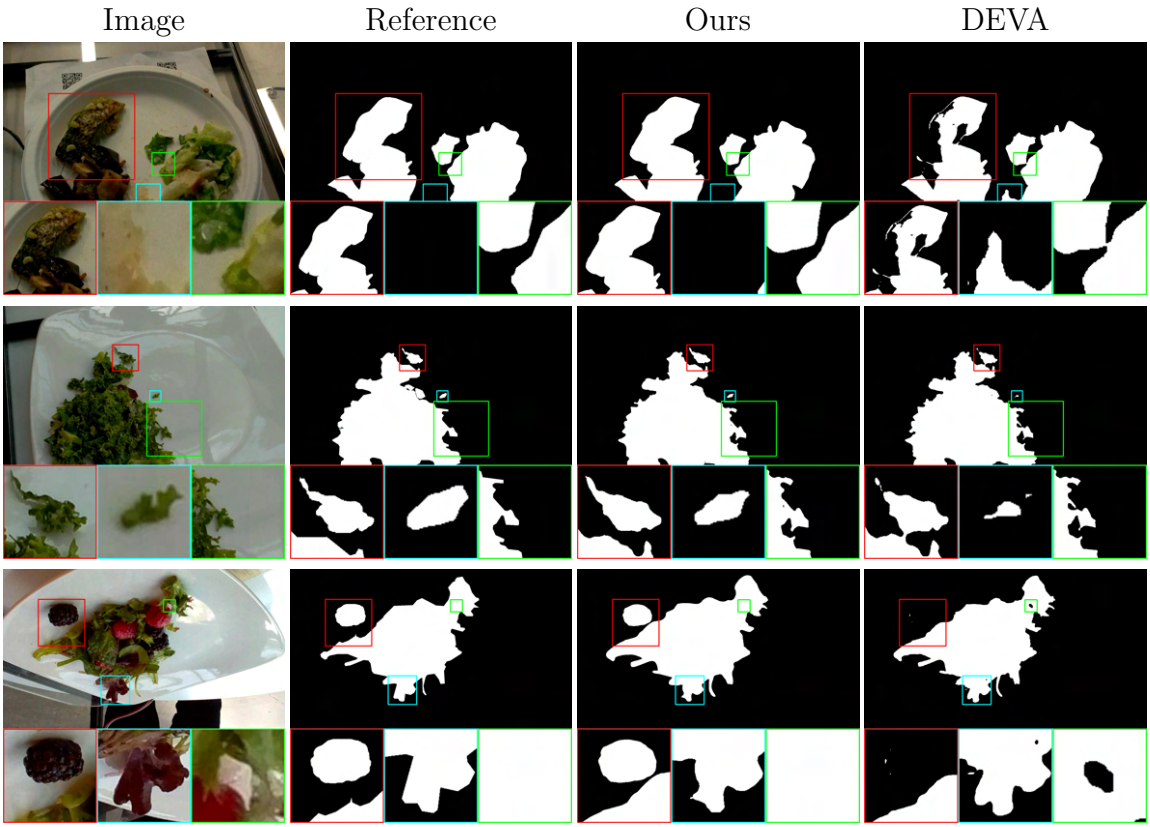


Figure 17: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by DEVA. The dataset used is Nutrition5k.

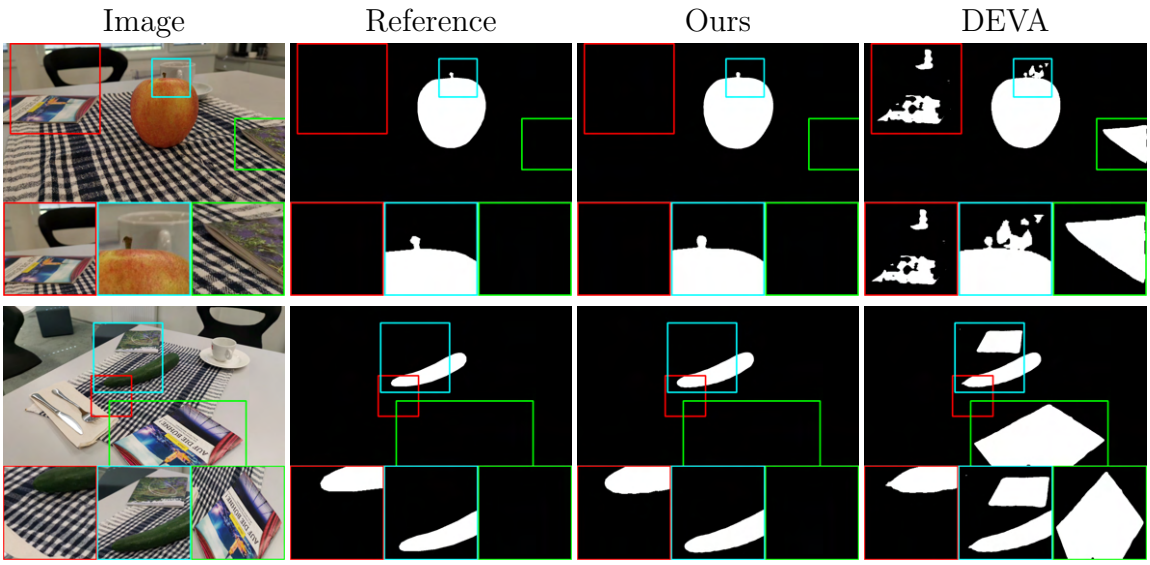


Figure 18: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by DEVA. The dataset used is Vegetables & Fruits.

DEVA shows very coarse segmentation, often including too many irrelevant elements in the mask and failing to capture the shape of the food accurately. The quality of the masks is inconsistent, sometimes missing large parts of the objects or including a lot of noise.

5.5.1.3 kMean++ and FoodMem

Figs. 19 and 20 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by kMean++. We applied Mozaic tool to do the visualizations attractive, as explained in Sec. 4.7.

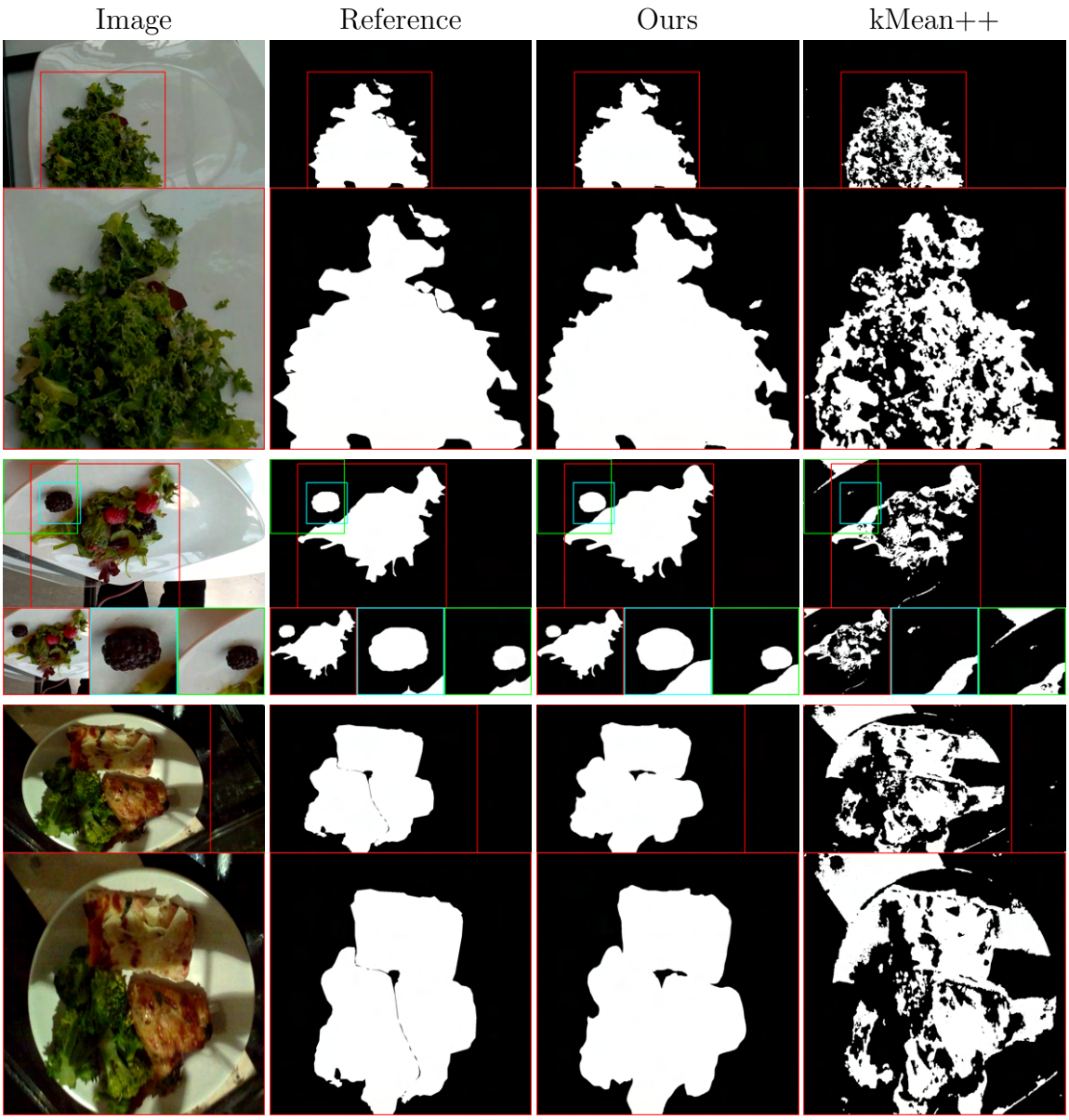


Figure 19: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by kMean++. The dataset used is Nutrition5k.

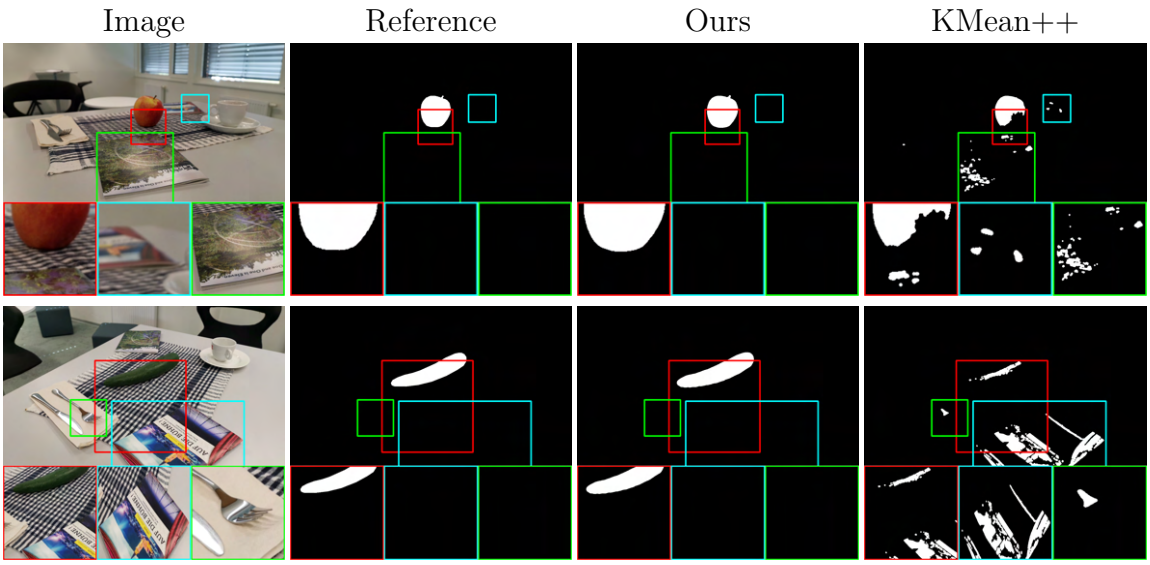


Figure 20: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by kMean++. The dataset used is Vegetables & Fruits.

kMean++ seems to have more variable performance, with some segmentations being a bit accurate and others being noisy or incorrectly shaped. Compared to FoodSAM and FoodMem, kMean++ tends to be less accurate and noisier.

5.5.2 Execution times comparison

We compare the execution time of the different models. Understanding the time each model takes to process video frames is important for evaluating their practical applicability, especially in real-time applications. Table 3 shows the average execution time each state-of-the-art method spent executing the videos from our dataset, as explained in Sec. 3.5.

Table 3: Average execution times of the different models. The models include FoodSAM, DEVA, kMean++ and our framework. The inference time were recorded in the format of hours:minutes:seconds.

Dataset	**Frames range**	**FoodSAM**	**DEVA**	**kMean++**	**Ours**
Nutrition5k	19-65	00:12:34	00:00:40	00:01:07	**00:00:25**
V&F	172-232	00:44:20	00:02:04	00:05:11	**00:00:31**

Fig. 21 shows a plot that relates the number of frames to the execution time in seconds of the dishes contained in our dataset for each state-of-the-art model.

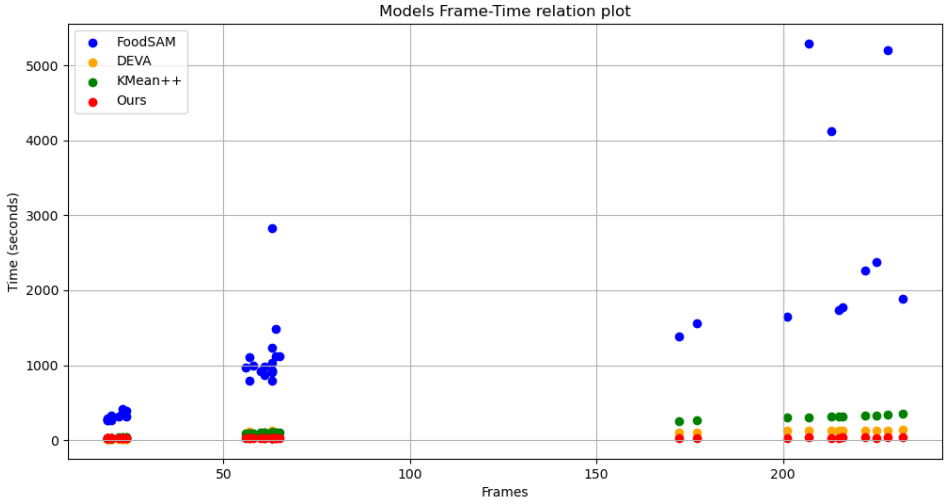


Figure 21: Models frame-time relation plot. The plot shows the models' average time (in seconds) to execute our dataset. Our framework is near 0 regardless of the number of frames.

Analyzing Table 3 and Fig. 21, the following conclusion on their performance over time can be reached:

- **FoodSAM:** Generally, the FoodSAM execution times are considerably longer than those of the other methods.
- **DEVA:** DEVA's execution times are relatively low compared to FoodSAM. This suggests that DEVA is efficient in terms of time, although previous analysis showed that its accuracy might be lower.
- **kMean++:** kMean++ shows consistent and generally low execution times, but like DEVA, its segmentation accuracy may not be the best.
- **FoodMem:** This method shows very low execution times, often less than a minute, indicating high efficiency in terms of time. This could make it attractive for applications requiring speed.

FoodSAM provides complete segmentation but at a significantly high time cost. FoodMem, on the other hand, is highly time-efficient, making it an attractive option when fast segmentation is required. If we consider the qualitative evaluation and the execution times, our framework is much more viable than FoodSAM since the masks become very similar for an incredibly short time.

5.5.3 Quality metrics evaluation comparison

We compare the two metrics explained in Sec. 5.3, mAP, and recall to evaluate the performance of the models. These two metrics are very important in image

recognition and segmentation. We try to find out in detail, through analysis, which of these models will offer the best performance and consistency for all possible food segmentation.

Table 4: Comparison of mean average precision (mAP) scores achieved by different models on two datasets: Nutrition5k and V&F. The models evaluated include FoodSAM, DEVA, kMean++, and our framework.

mAP				
Dataset	**FoodSAM**	**DEVA**	**kMean++**	**Ours**
Nutrition5k	**0.9192**	0.8825	0.4232	0.9098
V&F	0.8914	0.8548	0.4361	**0.9499**

Fig. 22 shows a plot that relates the mean average precision to the number of frames of the dishes contained in our dataset for each state-of-the-art model.

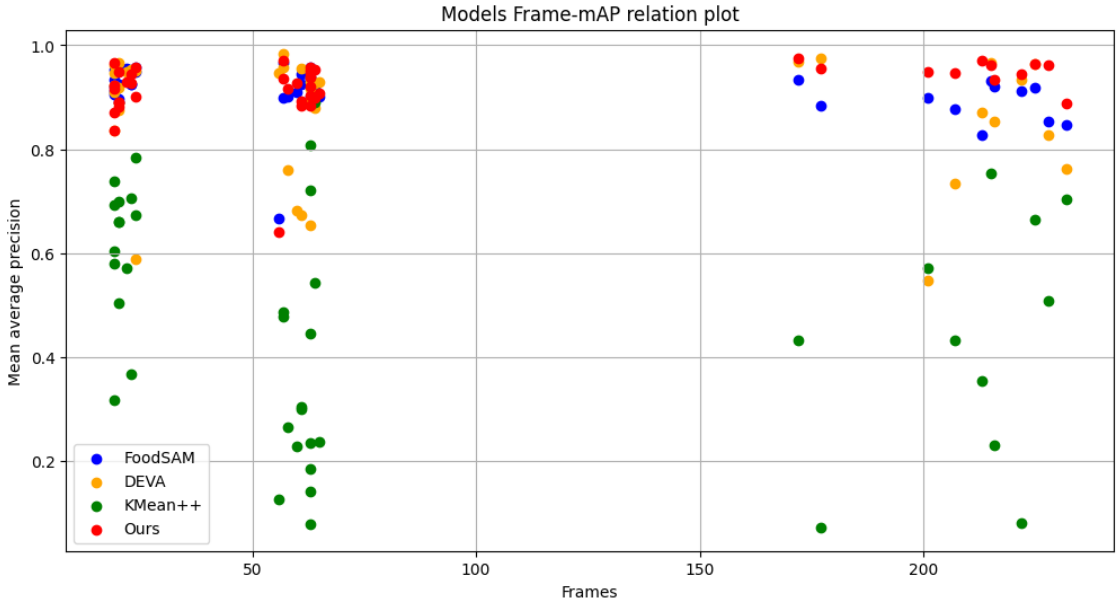


Figure 22: Models frame-mAP relation plot. The plot shows the mean average precision regarding the number of frames. Our framework is near 1.0, which means it performs well in detecting and segmenting objects accurately.

Table 5: Comparison of recall scores achieved by different models on two datasets: Nutrition5k and V&F. The models evaluated include FoodSAM, DEVA, kMean++, and our framework.

Recall				
Dataset	**FoodSAM**	**DEVA**	**kMean++**	**Ours**
Nutrition5k	**0.7752**	0.7301	0.6467	0.7708
V&F	0.9441	0.9328	0.9245	**0.9469**

Fig. 23 shows a plot that relates the recall to the number of frames of the dishes contained in our dataset for each state-of-the-art model.

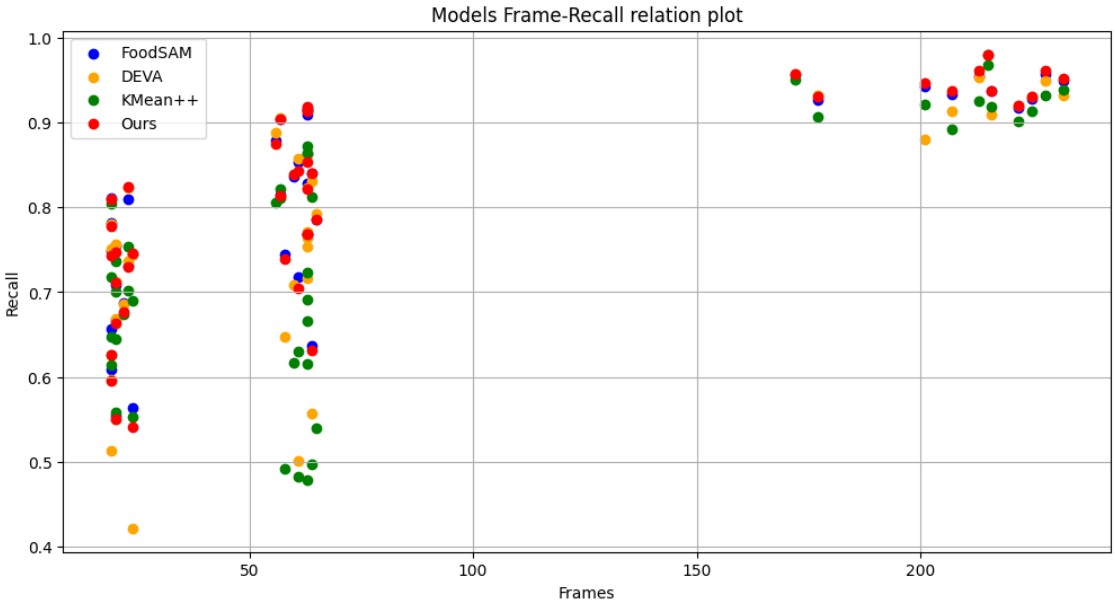


Figure 23: Models frame-recall relation plot. The plot shows the score regarding the number of frames. As seen, our framework recall scores are higher than the other methods. This means our framework better captures all relevant instances of the segmented objects.

Comparing FoodSAM, DEVA, kMean++, and our framework in terms of mAP and recall leads to significant conclusions about effectiveness and consistency:

- **FoodSAM and FoodMem:** These two methods have proven to be the most robust regarding mAP and recall. FoodMem, in particular, has shown extremely high mAP and recall values across nearly both datasets, indicating high average precision and a strong ability to retrieve many relevant instances correctly. FoodSAM closely follows, providing comparable and, in some cases, superior performance. The consistency of these methods across different datasets makes them highly reliable options for food segmentation tasks. We would like to point out that FoodSAM performs better than our framework in the Nutrition5k dataset. This is because FoodSAM was trained on datasets where the camera followed a predefined path to capture images, similar to the setup in the Nutrition5k dataset. On the other hand, our framework performs better in the Vegetables & Fruits dataset, where the camera has freedom of movement, resulting in less predictable image capture scenarios.
- **DEVA:** DEVA has also demonstrated solid performance, especially regarding mAP. Although it generally ranks slightly behind FoodSAM and our framework, it remains a viable option with competitive performance. However, in some scenes, its recall is lower, suggesting it might not be as effective at retrieving all relevant instances as the other two leading methods.

- **kMean++:** kMean++ has proven to be the least effective method among those compared, with significantly lower mAP and recall values across most datasets. This result suggests that kMean++ struggles to maintain precision and relevant instance retrieval at levels comparable to FoodSAM, DEVA, and our framework. Its inconsistent performance makes it a less preferable option for high-precision and recall applications.

FoodSAM and our framework stand out as the most reliable and effective options for food segmentation, providing high levels of precision and recall across various contexts. Now that we have collected all the qualitative and quantitative results, we can conclude that our framework stands out significantly in terms of performance and time compared to other methods.

5.6 Implementation settings

In implementing our framework, we configured the system to optimize performance and ensure smooth execution. Here is an overview of the key implementation settings:

- **GPU details:** We utilized an NVIDIA GeForce RTX 2080 Ti with 11GB of VRAM for accelerated processing of deep learning tasks. The powerful GPU architecture enabled rapid inference and efficient utilization of neural network models.
- **RAM:** Our system was equipped with with 32GB of DDR4 RAM, providing ample memory capacity for data processing, and caching of intermediate results. The generous RAM allocation ensures the smooth execution of memory-intensive tasks and minimizes the risk of memory-related bottlenecks.
- **Operating system (OS):** We operated on Microsoft Windows 11 Home, taking advantage of its modern features, enhanced security, and intuitive user interface for seamless software development and experimentation. Windows 11 provided a stable and user-friendly environment for running deep learning tasks and other computational workloads.
- **CPU:** The system featured an Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz, with 8 physical cores and 16 logical processors. This high-performance processor architecture enabled efficient parallel processing of computational tasks, contributing to overall system responsiveness and multitasking capabilities.
- **Storage:** Our system was supplemented with a TOSHIBA External USB 3.0 USB Device with a 1.81TB storage capacity. This external storage solution provided ample space for storing large datasets, model checkpoints, and experimental results. The high-speed USB 3.0 interface facilitated swift data transfer rates, ensuring efficient data access and management during the course of our experiments and development activities.

- **Software environment:** We utilized Python 3.8.19 as the primary programming language and deep learning frameworks such as PyTorch. Docker managed software dependencies and ensured reproducibility across different computing environments.
- **Development tools:** JetBrains IntelliJ IDEA acted as our primary integrated development environment (IDE), offering advanced code editing, debugging, and version control features. Git and GitHub were utilized for collaborative software development and version management, facilitating seamless code collaboration and sharing.

We created an optimized environment for developing and running our framework by configuring these implementation settings, ensuring efficient resource utilization, and maximizing performance across various computational tasks.

5.7 Ablation study

In this section, we conduct an ablation study to examine the effects of modifying the masks produced by SETR. In particular, we explore the changes that occur when increasing masks from 1 to 3, 6, or 9. Our objective is to provide quality and quantity results for the modifications made. We provide mask comparisons as a qualitative result, and we also analyze the execution time and evaluate their performance using mean average precision and recall metrics as a quantitative result.

5.7.1 Mask comparison

We display the segmentation masks produced by our framework with the settings modification stated in the previous section. This visual comparison helps illustrate the differences in segmentation quality and accuracy.

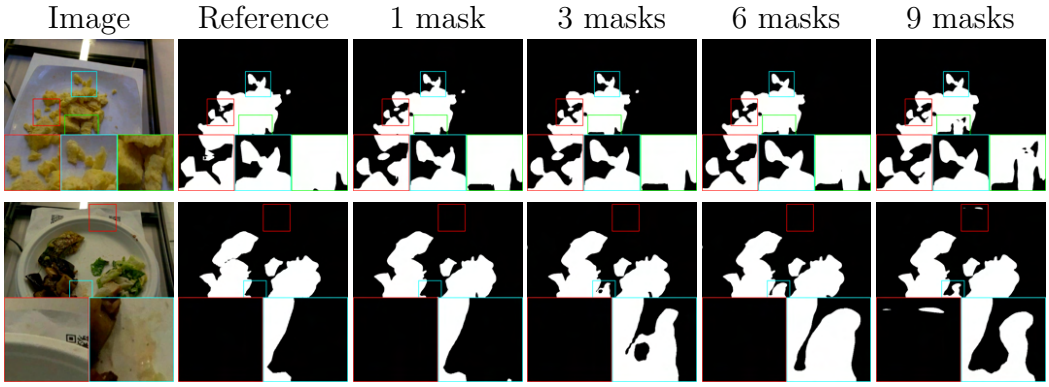


Figure 24: Comparison between original images, ground truth masks, 1 mask generated by SETR, 3 masks generated by SETR, 6 masks generated by SETR, and 9 masks generated by SETR. The dataset used is Nutrition5k.

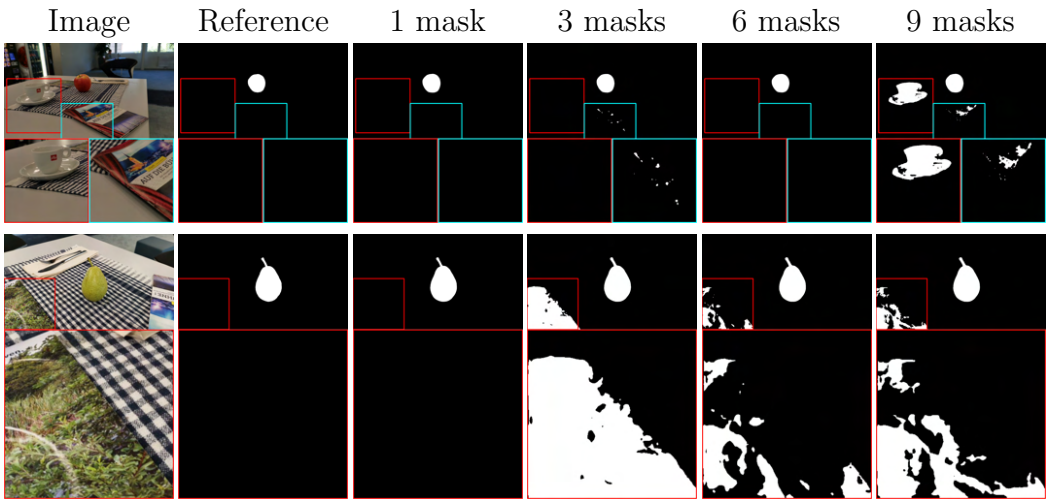


Figure 25: Comparison between original images, ground truth masks, 1 mask generated by SETR, 3 masks generated by SETR, 6 masks generated by SETR, and 9 masks generated by SETR. The dataset used is Vegetables & Fruits.

By comparing the masks generated by FoodMem with different settings (1 mask, 3 masks, 6 masks, and 9 masks generated by SETR) from Figs. 24 and 25, we got the following conclusion: As the number of masks generated by SETR increased, more segmentation errors appeared. This is mainly because the process tends to drag SETR's initial segmentation errors, compounding inaccuracies as the number of masks increases. The segmentation was more consistent and less prone to errors with fewer masks. In other words, using fewer masks resulted in more reliable and generalized segmentation.

5.7.2 Execution times comparison

In this section, we will compare the execution times of the modified settings of our framework. Understanding the time each set takes to process video frames is important for evaluating their practical applicability, especially in real-time applications. Table 6 shows the average execution time each setting modification spent executing the videos from our dataset, as explained in Sec. 3.5.

Table 6: Average execution times of our framework's different settings. The settings include 1 mask, 3 masks, 6 masks and 9 masks. The inference time were recorded in the format of hours:minutes:seconds.

Dataset	**Frames range**	**1 mask**	**3 masks**	**6 masks**	**9 masks**
Nutrition5k	19-65	**00:00:25**	00:00:35	00:00:50	00:01:05
V&F	172-232	**00:00:31**	00:00:39	00:00:45	00:00:56

Fig. 26 shows a plot that relates the number of frames to the execution time in seconds of the dishes contained in our dataset for each state-of-the-art model.

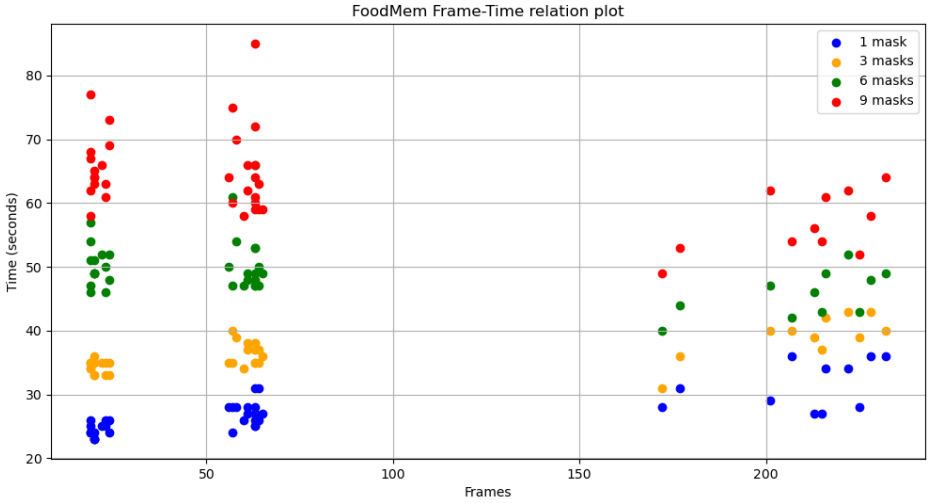


Figure 26: FoodMem's different settings frame-time relation plot. The plot shows the average time (in seconds) the model took to execute our dataset with different number of masks. As seen, 1 mask is nearer to 0 than the other settings.

As seen in Table 6 and Fig. 26, for both datasets, the execution time increases as the number of masks increases from 1 to 9. This pattern indicates that generating and processing more masks require more computational resources and time. Also, the increase in execution time is roughly proportional to the increase in the masks.

5.7.3 Quality metrics evaluation comparison

In this section, we compare the metrics explained in Sec. 5.3, mean average precision and recall, to evaluate the performance of the FoodMem's different settings. These two metrics are very important in image recognition and segmentation. We try to find out in detail, through analysis, which of these models will offer the best performance and consistency for all possible food segmentations.

Table 7: Comparison of mean average precision (mAP) scores achieved by different FoodMem settings on two datasets: Nutrition5k and V&F. The settings include 1 mask, 3 masks, 6 masks, and 9 masks.

mAP				
Dataset	**1 mask**	**3 masks**	**6 masks**	**9 masks**
Nutrition5k	**0.9098**	0.9025	0.9005	0.9082
V&F	**0.9499**	0.9027	0.9124	0.9050

Fig. 27 shows a plot that relates the mean average precision to the number of frames of the dishes contained in our dataset for each FoodMem's setting.

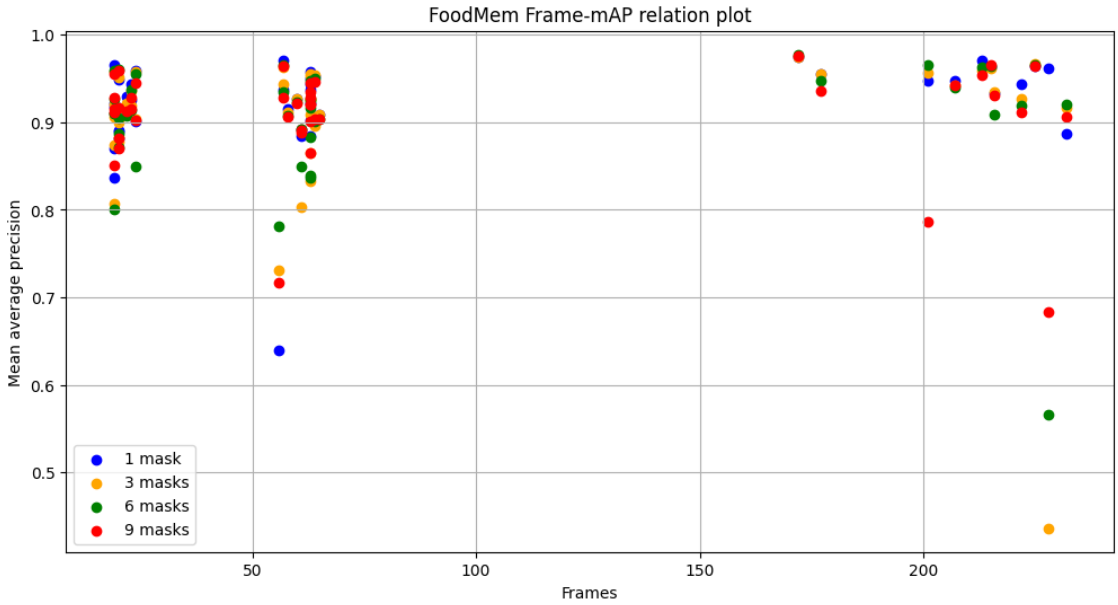


Figure 27: FoodMem's settings frame-mAP relation plot. The plot shows the mean average precision regarding the number of frames. As seen, 1 mask has higher values, which means it performs well in detecting and segmenting objects accurately.

Table 8: Comparison of recall scores achieved by different FoodMem's settings on two datasets: Nutrition5k and V&F. The settings include 1 mask, 3 masks, 6 masks and 9 masks.

Recall				
Dataset	**1 mask**	**3 masks**	**6 masks**	**9 masks**
Nutrition5k	**0.7708**	0.7688	0.7663	0.7690
V&F	**0.9469**	0.9419	0.9438	0.9430

Fig. 28 shows a plot that relates the recall to the number of frames of the dishes contained in our dataset for each FoodMem's setting.

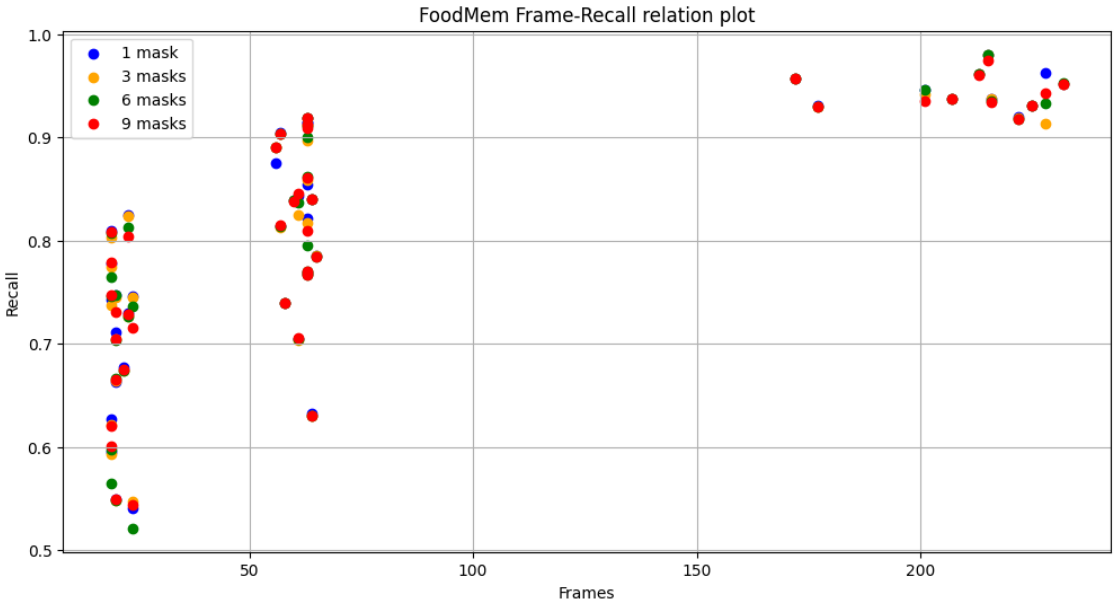


Figure 28: FoodMem's different settings frame-recall relation plot. The plot shows the recall regarding the number of frames. As seen, 1 mask has higher values, which means it better captures all relevant instances of the objects being segmented.

Comparing our framework's different settings in terms of mAP and recall leads to significant conclusions:

- Generating 1 mask is optimal for both mAP and recall, providing the best balance of precision and ability to recover relevant instances.
- Introducing more masks generally leads to a decline in both mAP and recall, likely due to the introduction of segmentation errors.

For optimal performance, considering the quality and quantitative results, generating 1 mask leads to the best segmentation results, as it does not independently drag SETR's errors at segmenting frames.

5.8 Limitations

This section outlines the possible limitations of our framework, helping to contextualize the findings and guide further research. Here is an explanation of the limitations:

- **Hardware and software dependencies:** The performance of our framework can be influenced by hardware specifications (e.g., GPU, CPU) and software dependencies (e.g., library versions).
- **Inherited SETR's training data limitation:** SETR has been trained on the FoodSeg103 dataset, which comprises 7118 images depicting 730 distinct

dishes. While this dataset provides a wide variety of food images for training, it may not encompass the full spectrum of possible dishes encountered in real-world scenarios. Consequently, our framework's performance may be limited when presented with food items or variations not adequately represented in the training data. This could lead to instances where certain foods are not segmented accurately or where objects resembling food are incorrectly classified as food items.

- **Inherited limitations from XMem++:** Our framework builds upon the model XMem++, inheriting its strengths and limitations. XMem++ itself may have constraints related to its algorithmic design, computational requirements, or performance characteristics. For example, limitations in memory usage or processing speed could impact the scalability or efficiency of our framework when applied to large datasets or high-resolution videos.

- **Lighting sensitivity and shadows:** Our framework's performance can be influenced by lighting conditions, particularly when the video is not correctly illuminated or too dark. Insufficient lighting may reduce visibility and contrast between food items and their backgrounds, affecting the algorithm's ability to detect and segment food objects accurately. As a result, our framework may not detect all food items or produce optimal segmentation results in low-light environments.

- **File format requirement :** Our framework currently requires video frames to be in JPG format and masks to be in PNG format. This limitation restricts the flexibility of input data and may necessitate preprocessing steps to convert files into the required formats.

6 Conclusions and future work

6.1 Conclusions

We summarize the study's key findings and results regarding our framework's performance and capabilities and provide a brief overview of the main takeaways and implications of the research. We highlight the successful integration of SETR and XMem++ to create our framework, emphasizing its effectiveness in achieving video semantic segmentation. Through the synergy of both models, our framework shows robust performance in accurately segmenting food items across multiple datasets. We discuss our framework's segmentation performance, emphasizing its accuracy, efficiency, and flexibility. We highlight its ability to efficiently process video streams while producing accurate segmentation results, making it suitable for real-world applications requiring food recognition and analysis. We acknowledge the limitations of our framework, such as its sensitivity to low-light conditions and potential challenges with recognizing less common food items. By addressing these limitations, we aim to provide a balanced assessment of our framework's capabilities and areas for improvement.

6.2 Future work

We identify some future research and development work that can enhance our framework's capabilities and address its limitations. This section helps us look ahead at what future investigations and improvements to our framework need.

- **Dataset expansion:** We propose expanding the training dataset to include a more diverse range of food items, thereby improving our framework's ability to accurately segment a wider variety of foods.
- **Enhanced low-light detection:** We suggest implementing robust low-light detection mechanisms to improve our framework's performance in challenging lighting conditions.
- **Adaptive scene memory:** We recommend enhancing XMem++'s scene memory capabilities to adapt to changing visual contexts, thereby improving segmentation accuracy.
- **Expansion of features:** Incorporating additional features such as volume estimation, calorie estimation, and panoptic segmentation can enhance our framework's utility and provide more comprehensive insights into food composition and nutritional content. These additional functionalities can extend our framework's applicability in dietary assessment, health monitoring, and culinary analysis.
- **Exploration of additional evaluation metrics:** Introducing more evaluation metrics beyond mAP and recall can provide a finer assessment of our

framework's performance. Metrics such as intersection over union (IoU), F1 score, and precision can offer insights into segmentation accuracy, boundary delineation, and overall model robustness. By incorporating a diverse set of evaluation metrics, we can better understand our framework's strengths and areas for improvement.

- **Support for diverse image extensions:** Enhancing our framework's compatibility by supporting a wider range of image extensions for video frames and masks can improve its usability and simplify data preparation processes. By accommodating formats such as JPG, JPEG, PNG, TIFF, and others, our framework can cater to address data sources and workflows.

By addressing these areas of future work, we aim to further enhance our framework's performance, versatility, and applicability in real-world scenarios, ultimately advancing the field of semantic food segmentation in video streams.

7 Appendix

This section serves to provide supplementary information on our framework. Sec. 7.1 offers more images about state-of-the-art models mask comparisons; Sec. 7.2 shows a collection of images expanding the ablation study results.

7.1 Supplementary validation material

We present additional material that supplements the main images discussed in Sec. 5.5.1.

7.1.1 FoodSAM and FoodMem

Figs. 29 and 30 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by FoodSAM. We applied the Mozaic tool to make the visualizations attractive, as explained in Sec. 4.7.

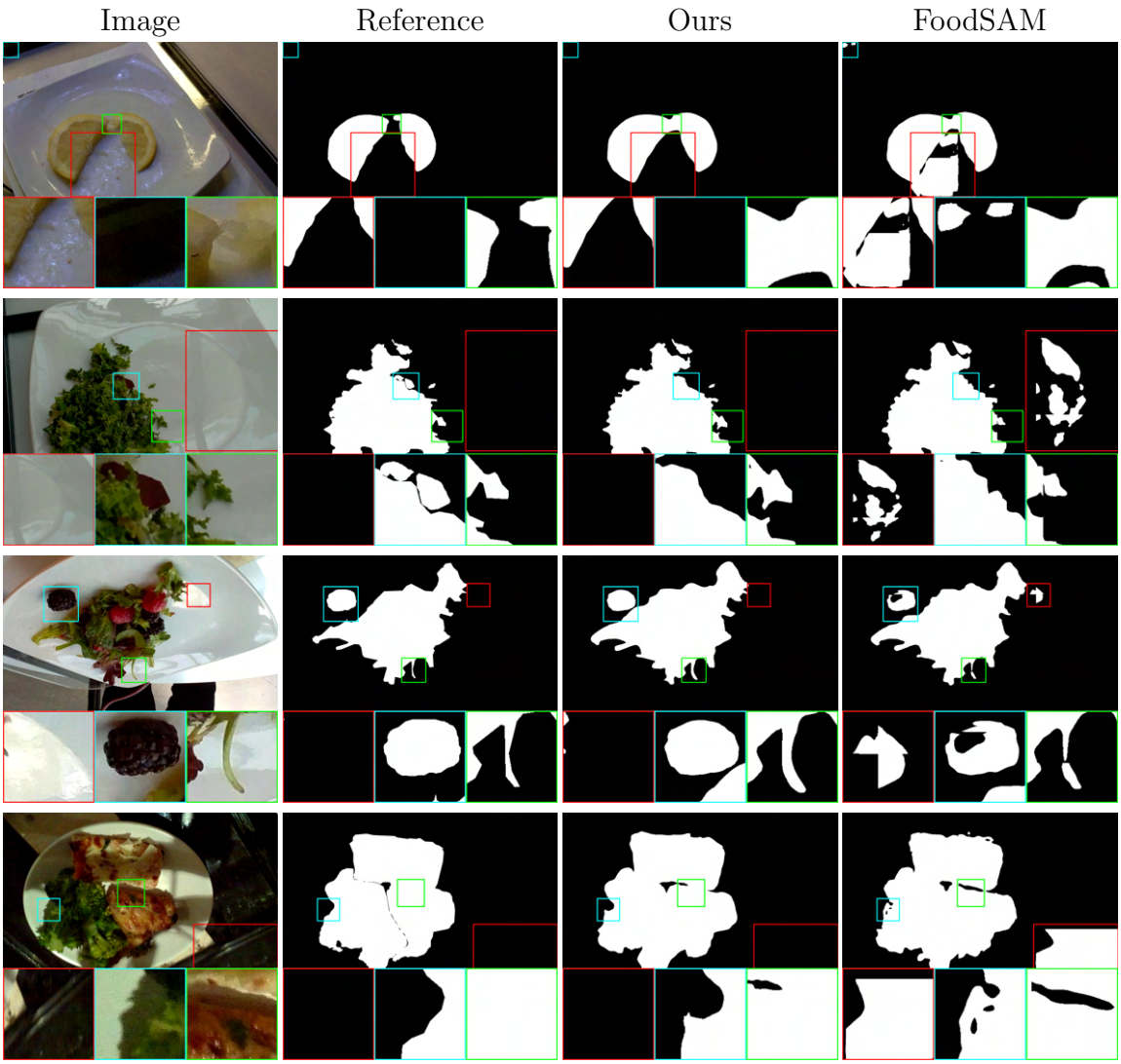


Figure 29: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by FoodSAM. The dataset used is Nutrition5k.

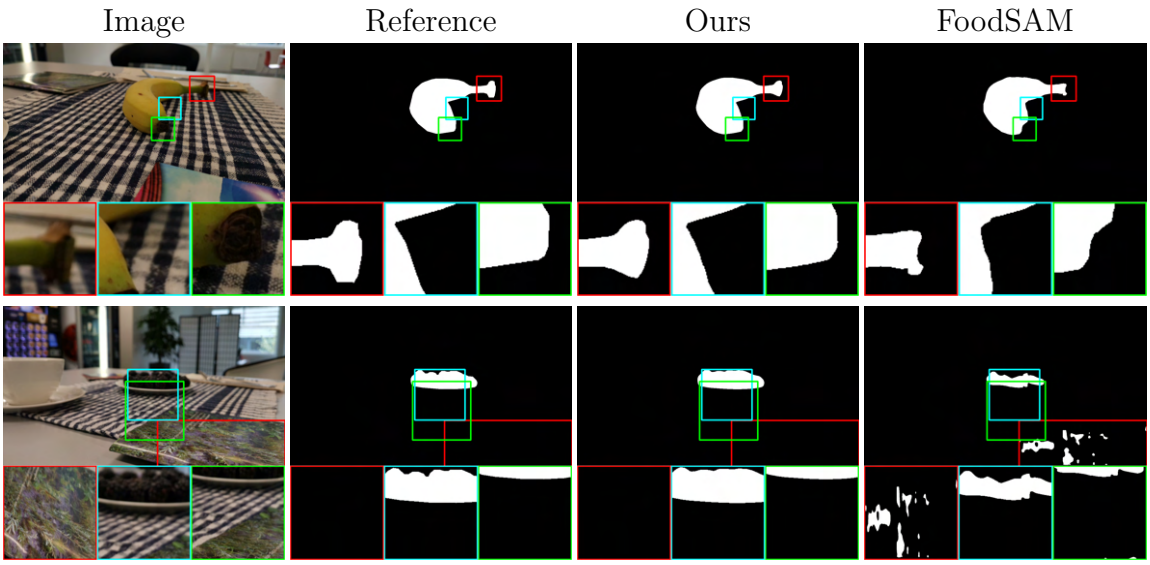


Figure 30: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by FoodSAM. The dataset used is Vegetables & Fruits.

7.1.2 DEVA and FoodMem

Figs. 31 and 32 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by DEVA. We applied the Mozaic tool to make the visualizations attractive, as explained in Sec. 4.7.

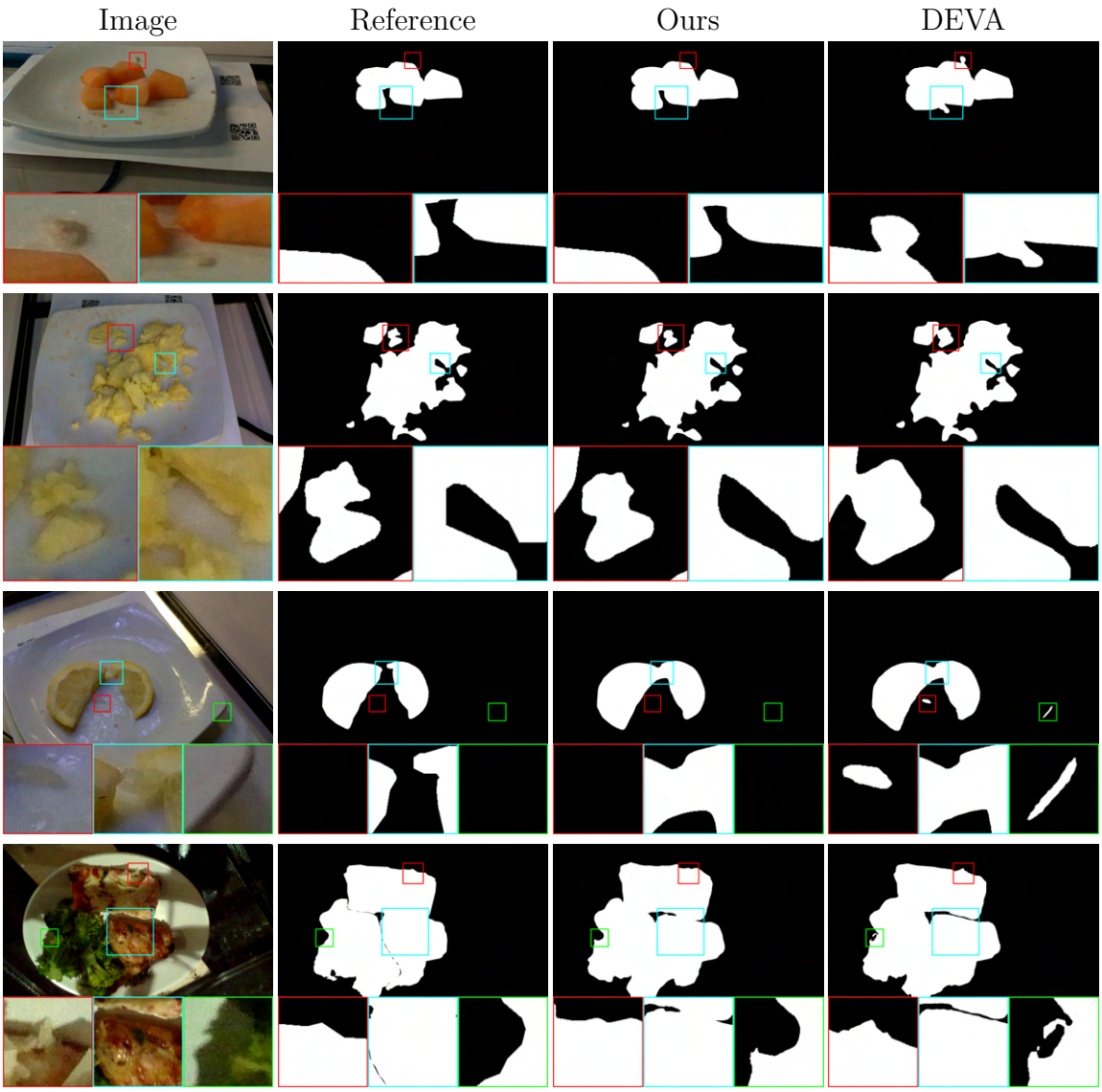


Figure 31: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by DEVA. The dataset used is Nutrition5k.

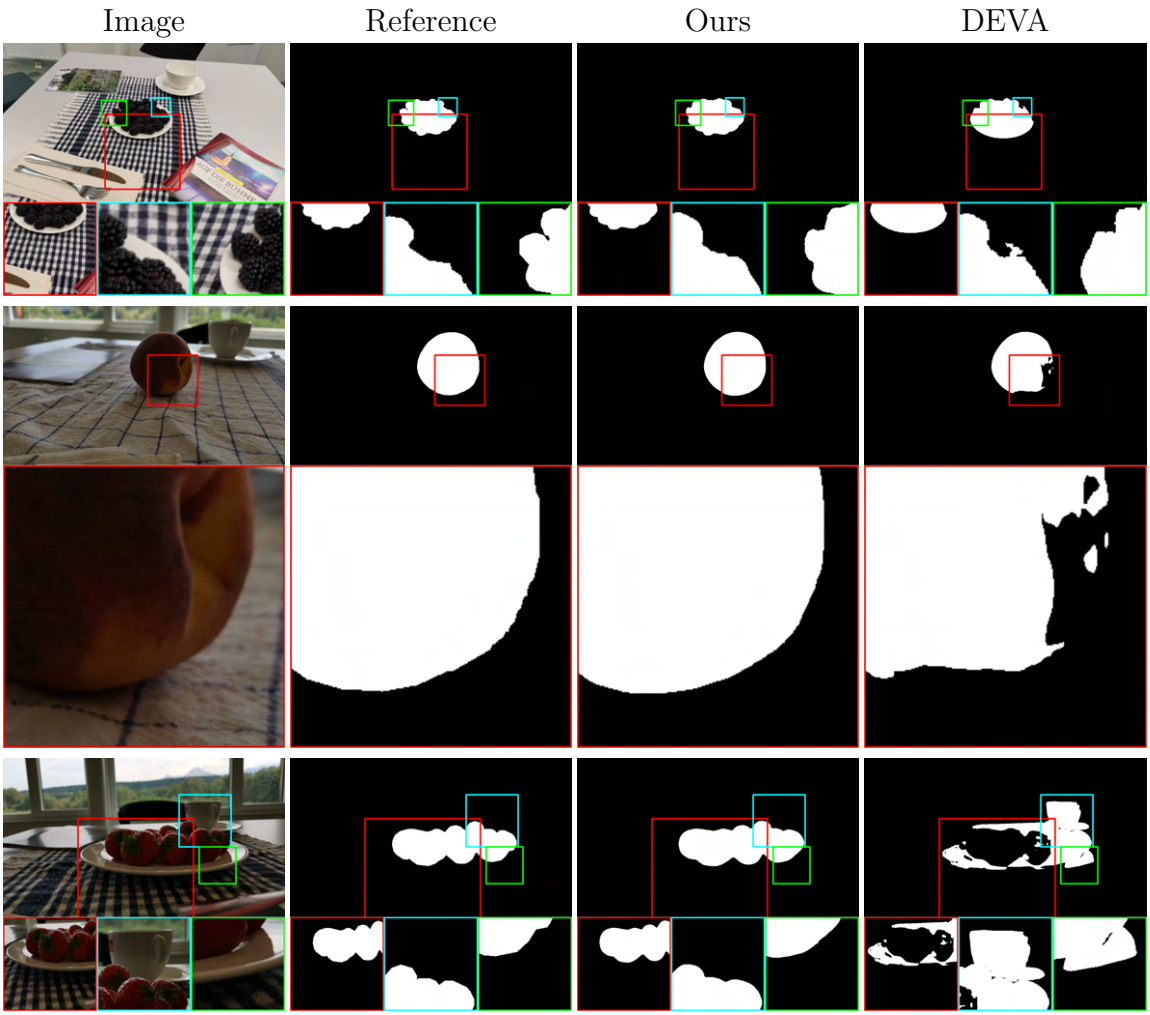


Figure 32: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by DEVA. The dataset used is Vegetables & Fruits.

7.1.3 kMean++ and FoodMem

Figs. 33 and 34 show the comparison between the original images, their ground truth masks, masks generated by FoodMem and masks generated by kMean++. We applied the Mozaic tool to make the visualizations attractive, as explained in Sec. 4.7.

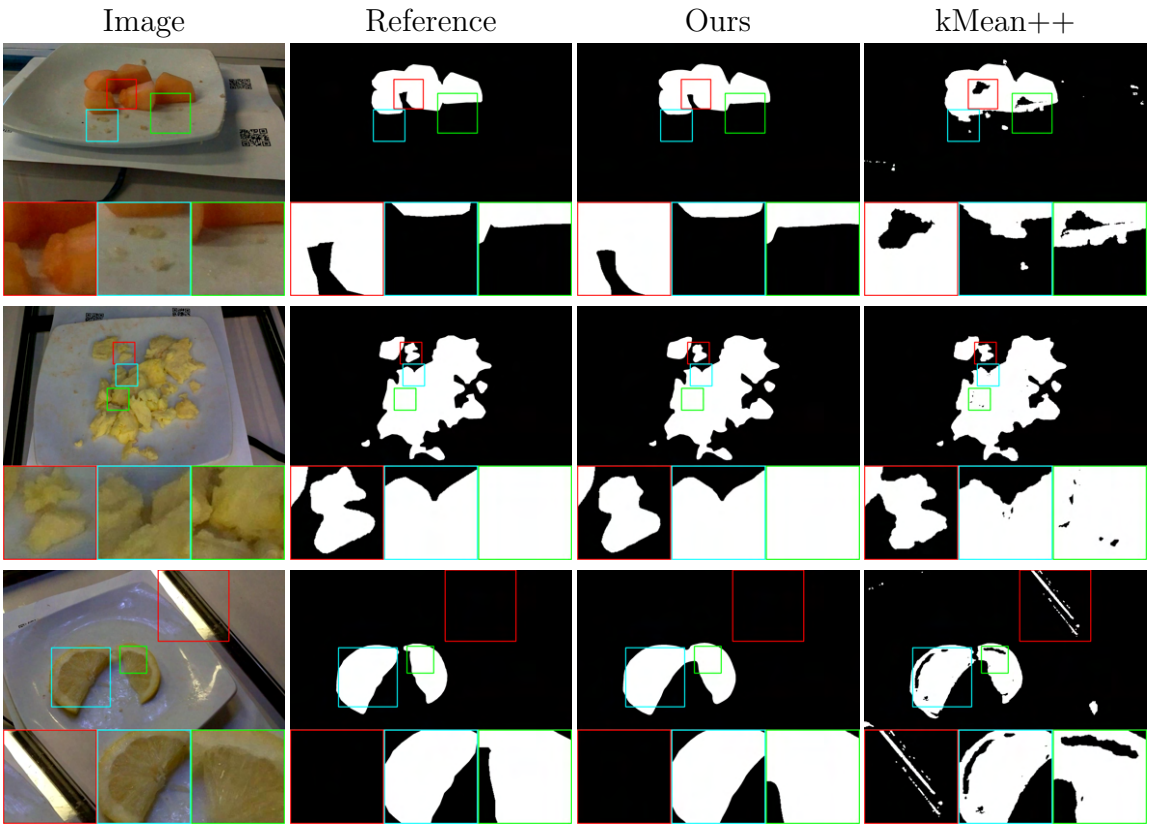


Figure 33: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by kMean++. The dataset used is Nutrition5k.

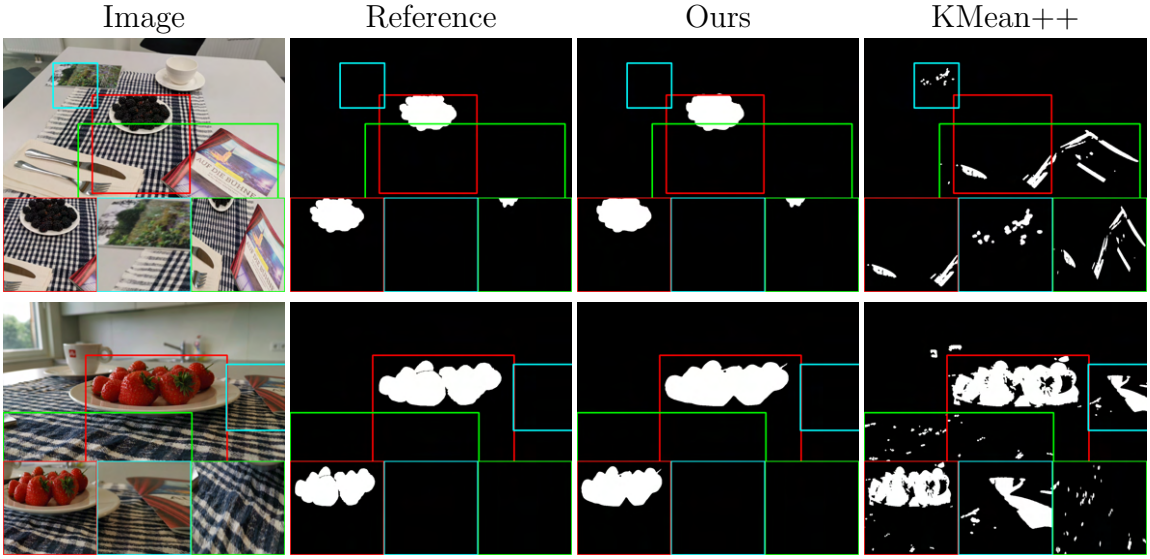


Figure 34: Comparison between original images, ground truth masks, masks generated by FoodMem, and masks generated by kMean++. The dataset used is Vegetables & Fruits.

7.2 Supplementary ablation study material

This section presents additional material that supplements the main images discussed in Sec. 5.7.1.

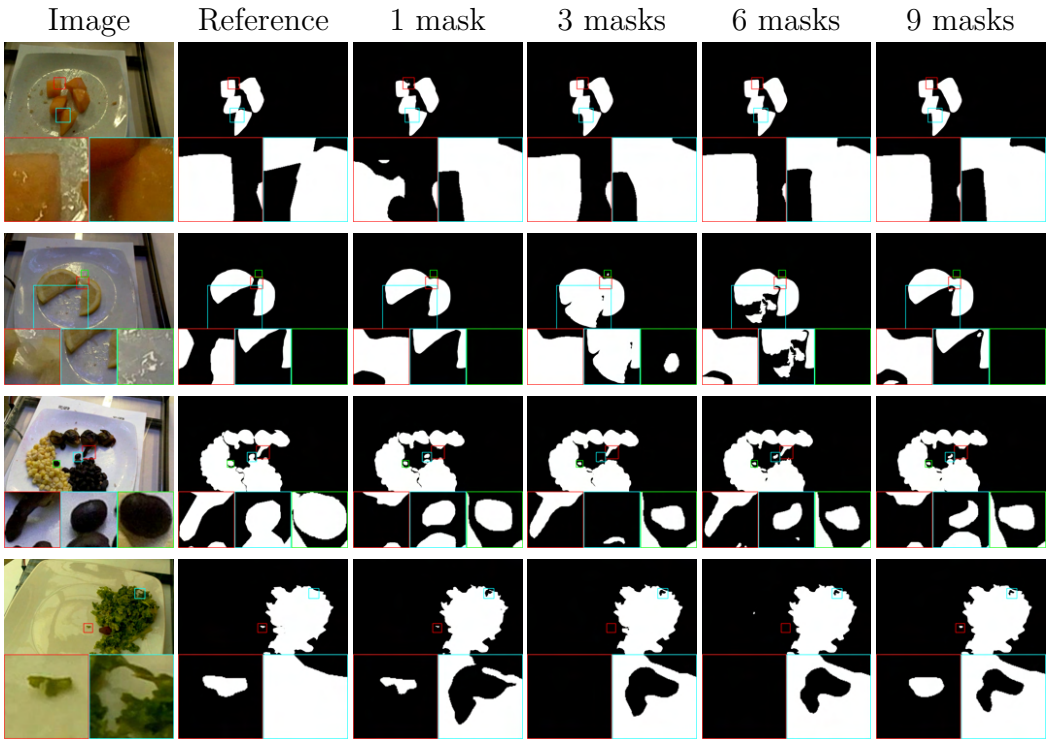


Figure 35: Comparison between original images, ground truth masks, 1 mask generated by SETR, 3 masks generated by SETR, 6 masks generated by SETR, and 9 masks generated by SETR. The dataset used is Nutrition5k.

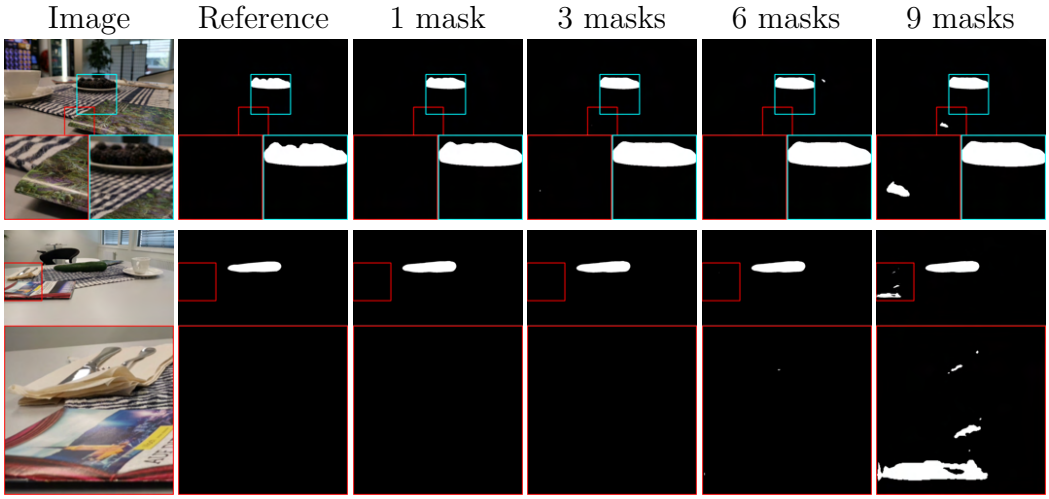


Figure 36: Comparison between original images, ground truth masks, 1 mask generated by SETR, 3 masks generated by SETR, 6 masks generated by SETR, and 9 masks generated by SETR. The dataset used is Vegetables & Fruits.

References

[1] S. Aslan, G. Ciocca, D. Mazzini, and R. Schettini, "Benchmarking algorithms for food localization and semantic segmentation," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 12, pp. 2827–2847, 2020.

[2] X. Lan, J. Lyu, H. Jiang, K. Dong, Z. Niu, Y. Zhang, and J. Xue, "Foodsam: Any food segmentation," *IEEE Transactions on Multimedia*, pp. 1–14, 2023.

[3] H. K. Cheng, S. W. Oh, B. Price, A. Schwing, and J.-Y. Lee, "Tracking anything with decoupled video segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 1316–1326.

[4] Y. A. Sari and A. Gofuku, "Measuring food volume from rgb-depth image with point cloud conversion method using geometrical approach and robust ellipsoid fitting algorithm," *Journal of Food Engineering*, vol. 358, p. 111656, 2023.

[5] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[6] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollar, and R. Girshick, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4015–4026.

[7] W. Zhang, Y. Zhou, Y. Wang, R. Wang, and H. Yang, "Automatic crack detection and segmentation of masonry structure based on yolov9-seg and edge detection," *Available at SSRN 4812249*, 2024.

[8] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," *CVPR*, 2022.

[9] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[10] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[11] Q. Thames, A. Karpur, W. Norris, F. Xia, L. Panait, T. Weyand, and J. Sim, "Nutrition5k: Towards automatic nutritional understanding of generic food," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 8903–8911.

[12] J. Steinbrener, V. Dimitrievska, F. Pittino, F. Starmans, R. Waldner, J. Holzbauer, and T. Arnold, "Learning metric volume estimation of fruits and vegetables from short monocular video sequences," *Heliyon*, vol. 9, no. 4, 2023.

[13] R. Yao, G. Lin, S. Xia, J. Zhao, and Y. Zhou, "Video object segmentation and tracking: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 4, pp. 1–47, 2020.

[14] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 1, pp. 87–110, 2022.

[15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[16] Meta, "Sam model design," https://segment-anything.com/, accessed: 2024-06-03.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[18] S. Gould, T. Gao, and D. Koller, "Region-based segmentation and object detection," *Advances in neural information processing systems*, vol. 22, 2009.

[19] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9404–9413.

[20] H. K. Cheng and A. G. Schwing, "Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model," in *European Conference on Computer Vision*. Springer, 2022, pp. 640–658.

[21] K. J. Malmberg, J. G. Raaijmakers, and R. M. Shiffrin, "50 years of research sparked by atkinson and shiffrin (1968)," *Memory & cognition*, vol. 47, pp. 561–574, 2019.

[22] M. Bekuzarov, A. Bermudez, J.-Y. Lee, and H. Li, "Xmem++: Production-level video segmentation from few annotated frames," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 635–644.

[23] D. Steinley, "K-means clustering: a half-century synthesis," *British Journal of Mathematical and Statistical Psychology*, vol. 59, no. 1, pp. 1–34, 2006.

[24] GCVCG, "Geometric computer vision and computational graphics group at university of barcelona," GitHub, 2022. [Online]. Available: https://github.com/GCVCG

[25] X. Wu, X. Fu, Y. Liu, E.-P. Lim, S. C. Hoi, and Q. Sun, "A large-scale benchmark for food image segmentation," in *Proceedings of the 29th ACM international conference on multimedia*, 2021, pp. 506–515.

[26] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*. Springer, 2012, pp. 421–436.

[27] P. Mishra and K. Sarawadekar, "Polynomial learning rate policy with warm restart for deep neural network," in *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019, pp. 2087–2092.

[28] T. Jain, C. Lennan, Z. John, and D. Tran, "Imagededup," https://github.com/idealo/imagededup, 2019.

[29] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, pp. 157–173, 2008.

[30] United States Department of Agriculture, "Myplate food groups, subgroups, and sample foods table," Flickr, 2017. [Online]. Available: https://www.flickr.com/photos/usdagov/36623517294

[31] G. Bradski, A. Kaehler *et al.*, "Opencv," *Dr. Dobb's journal of software tools*, vol. 3, no. 2, 2000.

[32] T. E. Oliphant *et al.*, *Guide to numpy*. Trelgol Publishing USA, 2006, vol. 1.

[33] S. Imambi, K. B. Prakash, and G. Kanagachidambaresan, "Pytorch," *Programming with TensorFlow: Solution for Edge Computing Applications*, pp. 87–104, 2021.

[34] R. Fox, *Linux with operating system concepts*. Chapman and Hall/CRC, 2021.

[35] C. Anderson, "Docker [software engineering]," *Ieee Software*, vol. 32, no. 3, pp. 102–c3, 2015.

[36] B. Hobbs and Y. Petit, "Agile methods on large projects in large organizations," *Project Management Journal*, vol. 48, no. 3, pp. 3–19, 2017.

[37] E. S. Hidalgo, "Adapting the scrum framework for agile project management in science: case study of a distributed research initiative," *Heliyon*, vol. 5, no. 3, 2019.

[38] M. O. Ahmad, D. Dennehy, K. Conboy, and M. Oivo, "Kanban in software engineering: A systematic mapping study," *Journal of Systems and Software*, vol. 137, pp. 96–113, 2018.

[39] S. P. Singh and G. Bhatnagar, "A robust image hashing based on discrete wavelet transform," in *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. IEEE, 2017, pp. 440–444.

[40] A. Bookstein, V. A. Kulyukin, and T. Raita, "Generalized hamming distance," *Information Retrieval*, vol. 5, pp. 353–375, 2002.

[41] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*. IEEE, 2020, pp. 237–242.

[42] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, "Localization recall precision (lrp): A new performance metric for object detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 504–519.

[43] L. N. Trefethen and J. Weideman, "The exponentially convergent trapezoidal rule," *SIAM review*, vol. 56, no. 3, pp. 385–458, 2014.

[44] K. Seers, "Qualitative data analysis," *Evidence-based nursing*, vol. 15, no. 1, pp. 2–2, 2012.

[45] J. Lau, J. P. Ioannidis, and C. H. Schmid, "Quantitative synthesis in systematic reviews," *Annals of internal medicine*, vol. 127, no. 9, pp. 820–826, 1997.