POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Segmenting Dynamic Objects in 3D from Egocentric Videos

Supervisors

Prof.ssa Tatiana TOMMASI

Dott.ssa Chiara PLIZZARI

Candidate

Francesco BORGNA

March 2024

Abstract

With the increasing availability of egocentric wearable devices, there has been a surge in first-person videos, leading to numerous studies aiming to leverage this data. Among these efforts, 3D scene reconstruction stands out as a key area of interest. This process allows for the recreation of the scene where the video was captured, providing invaluable support for the growing field of augmented reality applications. Some egocentric datasets include static 3D scans of recording locations, usually requiring costly hardware or dedicated scans. An alternative approach involves reconstructing the scene directly from video frames using Structure from Motion (SfM) techniques. This method not only captures the motion of the actor and the objects they interact with, including transformations (e.g., slicing a carrot) but also enables the use of any egocentric footage for scene reconstruction, even without physical access to the environment in real life. However, the task of decomposing dynamic scenes into objects has received limited attention. For example, SfM finds it challenging to distinguish between moving and static parts, resulting in cluttered point cloud reconstructions where the same object may appear superimposed or in multiple places within the scene.

In this thesis, we combine SfM with egocentric methods to segment moving objects in 3D. This is achieved by creating a scene with COLMAP, a SfM algorithm, and then modifying a recent algorithm called NeuralDiff, originally designed for producing 2D segmentations of static objects, foreground, and actors, to extract 3D geometry. Additionally, we explored ways to reduce the overall computational demands, such as by simplifying the NeuralDiff architecture to better meet our goals by merging the foreground and actor streams, and by developing an intelligent video frame sampling technique that captures the essence of the scene using fewer frames.

Acknowledgements

Grazie Mamma e Papà,
e a tutti quelli che mi sono stati vicino.

Table of Contents

List of Tables VI

List of Figures VIII

Acronyms XVI

I Related Works 6

1 Photogrammetry 7
- 1.1 Photogrammetry Fundamentals . . . 7
- 1.2 Structure from Motion: SfM . . . 9
  - 1.2.1 Feature Detection and Matching . . . 10
  - 1.2.2 Estimating fundamental and essential matrices . . . 13
  - 1.2.3 Estimating camera pose from essential matrix . . . 15
  - 1.2.4 Multi-view Structure from Motion . . . 16
  - 1.2.5 COLMAP . . . 16
  - 1.2.6 Monocular Depth Estimation . . . 19

2 Neural Rendering 22
- 2.1 Previous works . . . 22
- 2.2 NeRF: Representing scenes as neural radiance fields for view synthesis 24
  - 2.2.1 NeRF . . . 24
  - 2.2.2 Results . . . 28
- 2.3 NeuralDiff: segmenting 3D objects that move in egocentric videos . 32
  - 2.3.1 NeuralDiff . . . 33
  - 2.3.2 EPIC-Diff benchmark . . . 36

3 Datasets 40
- 3.1 EPIC-KITCHENS . . . 40
  - 3.1.1 Motivation . . . 40

3.1.2 Data collection . . . 41
3.1.3 Data Annotation pipeline . . . 42
3.1.4 Benchmarks and baseline results . . . 44
3.2 EPIC-KITCHENS 100 . . . 48
3.2.1 Motivation . . . 48
3.2.2 Data collection . . . 48
3.2.3 Annotation . . . 49
3.3 EPIC-Fields . . . 50
3.3.1 Data . . . 51
3.3.2 Benchmarks . . . 53
3.4 VISOR . . . 56
3.5 Other egocentric datasets . . . 57
3.5.1 Ego4D: Around the World in 3,000 Hours of Egocentric Video 57
3.5.2 Aria Digital Twin: a new benchmark dataset for egocentric 3D machine perception . . . 58

II Our Contribution 59

4 Methodology 60
4.1 Goals . . . 60
4.2 Pipelines . . . 61
4.3 Filtering . . . 66

5 Experiments 69
5.1 Data selection . . . 69
5.2 Metrics . . . 70
5.2.1 PSNR: Peak signal-to-noise ratio . . . 70
5.2.2 AP: Average Precision . . . 71
5.3 COLMAP Reconstruction . . . 72
5.3.1 Results . . . 72
5.4 Monocular Pipeline . . . 75
5.5 Sampling Frames . . . 79
5.6 NeuralDiff Pipeline . . . 80
5.7 NeuralCleaner . . . 90

III Conclusions 93

6 Conclusions 94

Bibliography 96

List of Tables

3.1 Comparative overview of relevant datasets (action classes with > 50 samples) 41

5.1 Comparison of recostruction of scene P01_01 using different resolutions. The higher the resolution, the better. Too low resolution, as 114x64 in this case can lead to a unsuccessful reconstruction. 73
5.2 Comparison of recostruction details for scene P01_01 using different initial frames at same resolution of 228x128. The higher the frames, the better the reconstruction but at a higher computational cost. The columns' names are in the form "P01_01_xx" where xx represent the subsample, e.g. P01_01_08 is the subsample of scene P01_01 having 2598 initial frames. 73
5.3 **Split∼ 1000.** Number of frames resulting from the different sampling steps. In particular from the original number of frames (Original) are reduced with the homography filter to remove redundancy and keep overlap, resulting in Sampled. The reconstructed frames are the ones which were successfully reconstructed by COLMAP (Reconstructed). The obtained ones are the reconstructed frames filtered again with the homography filter (Obtained). The final samples (Int/Unif Samples) are the reconstructed frames without the previously obtained ones. The thresholds reported are referred to the last homography filter step. 79
5.4 **Split∼ 700.** 79
5.5 **NeuralDiff Pipeline Results on P01-01 at 114x64.** For the same scene P01-01 results of NeuralDiff pipeline trained on different amount of frames are reported. The Frames are selected using the three different sampling strategis: Intelligent, Uniform and AU (see Section 4.3). The frames are all at a 114x64 resolution. 81
5.6 NeuralDiff models trained on different scenes at ∼1000 frames, resolution 114x64. The column Improv. represents the difference between the previous column of the Intelligent split and the Uniform one. . . 86

5.7 NeuralDiff models trained on different scenes at ∼700 frames, resolution 228x128. The column I-U represents the difference between the previous column of the Intelligent split minus the Uniform one. 86
5.8 Metrics comparing the profile of the histograms of frames and annotations. In particular higher values of Cosine similarity indicates similarity; while the value of the two divergences represents the distance between the two distributions. 91
5.9 Results for NeuralCleaner, compared with the durations of the NeuralDiff pipeline. We can see that the durations are on average shorter of ∼30%. 91

List of Figures

1.1 **Pinhole Camera.** A world's object is captured by the camera making light pass through the pinhole and is then projected on the focal plane upside-down. 8

1.2 **Reference systems.** An example of different reference systems involved in a photogrammetry problem. In red is reported the world system while in black is the camera one. The yellow plane corresponds to the camera focal plane. 9

1.3 **Basic SfM Scenario.** Two cameras capture the same object from different viewpoints. 9

1.4 **Feature Detection Example** [20]. The easier patches to match to the background are the ones with sharp corners. 11

1.5 **SIFT features position.** Example of interest points, we can see that most of them are placed around corners or edges. 12

1.6 **Features correspondence** computed using BruteForce Matcher on a sample image. 13

1.7 **Point correspondence geometry.** In particular, are reported: the *epipoles* e, e'; the *epipolar plane*, which is any plane containing the line that connects the two camera centers, also known as *baseline*; the *epipolar line* which is the intersection of any epipolar plane with the image plane. 14

1.8 A point X viewed from two cameras with x_l being the distance of X from C_l and x_r being the distance of X from C_r. p_r and p_l are instead the projection of X on the respective camera's focal plane. . 15

1.9 **Multiple View Scenario.** In multiple view scenario multiple cameras are present, each one recording the scene from a different point of view. 17

1.10 **Building Rome in one day.** Result of Rome with 21K registered out of 75K images. 17

1.11 Top: input images. Middle: Inverse depth maps predicted by the presented approach. Bottom: corresponding point clouds rendered from a novel viewpoint. (Taken from [13]) 21

2.1 **NeRF.** Optimization of a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. The 2D novel views are obtained thanks to classic volume rendering techniques. Here in this example, given 100 images acquired from different viewpoints, they sample two novel views. 24

2.2 F_Θ **Scheme.** The input position **x** pass through 8 Fully connected (FC) layers of 256-channels. Each FC layer is followed by a ReLU activation function. This intermediate result is then concatenated with the input direction (**d**) and fed to one last FC with 128 channels that feed its output to a ReLU function. The output of the ReLU is the color **c** and the volume density (σ). 25

2.3 Here we reported the results obtained with different strategies, as written underneath each image. In particular, removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the Positional encoding instead, we obtain a blurred image, meaning that high frequencies are not captured nor represented. 25

2.4 Example of rays passing through an image plane of size 3x3 pixels. . 26

2.5 Probability Density Function of normalized coarse weights $\hat{w}_i$ along a ray with N_c samples. 28

2.6 Comparison of test images from the newly introduced synthetic dataset. The algorithms compared are NeRF, Local Light Field Fusion LLFF [46], and Scene Representation Network SRN [47]. The NeRF method can recover fine details in both geometry and appearance. LLFF exhibits some artifacts on the microphone and some ghosting artifacts in the other scenes. SRN produces distorted and blurry rendering for every scene. Neural Volumes struggle to capture details we can see from the ship reconstruction. 30

2.7 Comparison on the test set of the real images. As expected LLFF is performing pretty well being projected for this specific use case (forward-facing captures of real scenes). However, NeRF can represent fine geometry more consistently across rendered views than LLFF as we can see in Fern's and T-rex. NeRF is also able to reproduce partially occluded scenes as in the second row. SRN instead completely fails to represent any high-frequency content. 31

2.8 Given an egocentric video with camera reconstruction, NeuralDiff, a neural architecture, learns how to decompose each frame into a static background and a dynamic foreground, which includes every object that sooner or later will move and the actor's body parts. Each of these streams is learned by exploiting the characteristics of the scene that is going to be captured. Being a neural radiance field, NeuralDiff is also capable of rendering images from novel viewpoints as can be seen in the bottom right part of the scene. 33
2.9 Examples of frames with their corresponding manually segmented binary pixel-wise masks. 37
2.10 Three Scenes reconstruction from NeRF, NeRF-W and NeuralDiff, NeuralDiff+C+A. It can be seen that NeuralDiff produces less ghosting artifacts, captures most moving objects and shows more details, especially the *upgraded* version. 38
2.11 **Segmentation Masks.** Here the masks for which NeuralDiff scored best (top 4 rows) and worst (last 2 rows) are reported. 39

3.1 **Top** (left to right): time of day of the recording, a pie chart of high-level goals, histogram of sequence durations and dataset logo; **Bottom**: Wordles of narrations in native languages (English, Italian, Spanish, Greek and Chinese). Figure from [14]. 42
3.2 Narration Guidelines are given to each participant to be followed after the completion of a recording. Figure taken from [14] 43
3.3 Extracts from 6 transcription files in .sbv format. Figure from [14]. 43
3.4 Example of verbs and nouns classes clustered in wider groups. Figure taken from [14] 44
3.5 **From Top:** Frequency of verb classes in action segments; Frequency of noun clusters in action segments, by category; Frequency of noun clusters in bounding box annotations, by category; Mean and standard deviation of bounding box, by category. From [14] 45
3.6 Sample of consecutive action segments with keyframe object annotations from [14] 46
3.7 Sample qualitative results from the challenge's baseline of the Action Recognition Task 46
3.8 Sample qualitative results from the challenge's baseline of the Action Anticipation Task 47
3.9 Sample qualitative results from the challenge's baseline of the Object Detection Task 47
3.10 Annotation pipeline from [66]: **a** narrator, **b** transcriber **c** temporal segment annotator and **d** dependency parser. Red arrows show AMT crowdsourcing of annotations. 49

3.11 Comparing non-stop narrations (blue) to "pause-and-talk" narrations (red). Right: timestamps (dots) and segments (bars) for two sample sequences. "pause-and-talk" captures all actions including short ones. Black frames depict missed actions. Figure from [66]. 51

3.12 **Dynamic New View Synthesis**. Example from [68] of the output for the three different methods used: NeRF-W, T-NeRF+ and NeuralDiff. We can see how the initial labeling of difficulty for frames was accurate as the reconstructions struggle with Hard frames. 54

3.13 **UDOS.** Comparison,reported from [68], of the different methods' output. The 2D-based performs very well on dynamic objects, while 3D methods struggle a bit but can detect even semi-static objects. . 55

3.14 **VOS.** Comparison of the two different methods taken from [68]. The 3D method is better at having as output something close to the ground truth. The 2D method instead is performing poorly. 56

3.15 **VISOR Annotations and Benchmarks.** Sparse annotations of the P06-03 scene, where flour becomes dough in the end. Each color represents a different object. In the bottom part, the three proposed challenges are reported for the current scene. Namely: Semi-Supervised Video Object Segmentation (VOS), Hand Object Segmentation (HOS) and Where Did This Come From (WDTCF). Figure taken from [73]. 57

3.16 **Aria Glasses** exploded in its components. 58

4.1 Reconstruction of a pizza preparation video. The top and bottom frames give us a glance at the action performed during the video while the central pointcloud highlights the problems of SfM in dynamic environments like the superimposition of the same object on itself or the reconstruction of objects that are not always present in the scene, e.g. the two pizzas. 61

4.2 **Basic Pipeline.** In the basic pipeline a video (represented by the image of a person cooking) is reconstructed via COLMAP [15] and then fed to *NeuralDiff* [11]. At this stage, the frames are decomposed into actor, foreground, and background (as can be seen in the frames reported below *NeuralDiff*). A clean reconstruction is obtained by running another COLMAP step on the extracted background frames. 62

4.3 **Monocular Pipeline.** The monocular pipeline shares the first part with the basic pipeline. A video is reconstructed via COLMAP and fed to *NeuralDiff*. The difference is that here the dynamic layers are projected in 3D and points closer than a distance ς are segmented as dynamic. The red dots represent the pixel projected in the scene in the top image while in the bottom represents the dynamic labeled points of the COLMAP reconstruction. 64

4.4 **Overview of NeuralDiff.** Given only the camera viewpoint g_t and a frame specific code z_t^f (learned latent variable), the *NeuralDiff* three stream architecture learns to predict the color value of pixel x_{ut} by combining information coming from the static model of the background, and from two dynamic components, one for the foreground objects, and one for the actor. Figure reported from [11]. 64

4.5 **NeuralDiff Pipeline.** In this pipeline we obtain the three motion layers as in the other methods but the segmentation is performed by querying the static neural renderer with the positions of the points belonging to the COLMAP reconstruction. Each point is segmented as a dynamic if its density is less than a predefined value. 66

4.6 **NeuralCleaner Architecture.** Given only the camera viewpoint g_t and a frame specific code z_t^f (learned latent variable), the NeuralCleaner two-stream architecture learns to predict the color value of pixel x_{ut} by combining information coming from the static model of the background, and from the dynamic component, which is unique in this case. The parameters of this model are learned using a probabilistic loss $\mathcal{L}$. 67

4.7 Sampling Steps in our pipelines. The initial video is subsampled by EF-Sampling and the resulting frames are fed to COLMAP. Once COLMAP reconstructed the scene these frames are again subsampled via Intelligent Sampling and fed to *NeuralDiff*. 68

4.8 Example of overlapping frames (X and Y). The dot represent the features extracted. The left and central examples present the same level of overlap even though the image frames are closer together in the central example because the number of features shared is the same. The right examples instead present a higher level of overlapping due to the bigger number of features. 68

5.1 Two frames from EPIC-Diff and their corresponding manual dynamic (in white)/background (in black) segmentation masks. 70

5.2 Example of Precision-recall curve. We can see how the bottom line model represents the worst a model can perform, e.g. predict every sample as it is coming from the same class,if the dataset is balanced. A better model would *tend* to the upper-right corner, which instead represents the best possible model, a model that have maximum precision and recall. 71

5.3 Different COLMAP pcd reconstructions changing number of samples, that is reported after the @. Each row is the same reconstruction viewed from different viewpoints. From top to bottom, the number of frames increases. We can see how the number of frames positively affects the reconstruction. 74

5.4 Different COLMAP pcd reconstructions changing resolution. Each row is the same reconstructions viewed from different viewpoints. The first reports the reconstruction for a resolution of 456x256 and the bottom one the half resolution. It is clear how the resolution has a beneficial impact on the overall reconstruction. 75

5.5 COLMAP reconstructions for scenes P03-04, P04-01, P09-02, P16-01,P21-01. Each row represents a different kitchen. 76

5.6 Different scenes where monocular depth estimation was performed. In particular on the right we can find the frame that is instead projected (in red) on the left in the 3D reconstruction of that kitchen. The red frustum (pyramid) represents the camera position and orientation in the space. The name on top is in the form "Scene:@x" where scene represents the kitchen and x represents the number of the frame. 77

5.7 Dynamic points segmented on scene P01-01 using Monocular Pipeline. 78

5.8 Different Segmentation changing the distance threshold. Each point is segmented as dynamic if its distance from a pixel projected in 3D space is less than a threshold Th. The scene is P03-04 and in the left side we can find the static part while on the right side, we have the dynamic points. Here we can notice how this method is pretty inaccurate. 78

5.9 Visualization of the sampling of scene P01-01 for the three different methods: Intelligent, Uniform and AU using 217 frames in total. The left box is a proposal we gave to visualize how the frames actually spread along the temporal axis, where a line is drawn in correspondence with each sample. The right boxes represent instead histograms with the frequencies of the samples on the entire duration of the video. 82

5.11 Qualitative results for the static reconstruction of P01-01 scene at 217 frames. In red is highlighted a dynamic plate, while in green is a dynamic pan. We can see that Intelligent sampling is correctly removing the objects while Uniform can not. 83
5.12 Qualitative results for the static reconstruction of P03-04 scene. In red is highlighted a dynamic can that is successfully removed in Intelligent sampling while not in Uniform. 84
5.13 Comparative of the qualitative results for different samplings of the P01-01 scene. 85
5.14 Qualitative results of different kitchens. On the first row is reported the COLMAP reconstruction while below is the corresponding cleaned pointcloud. 87
5.15 Comparison of frequencies for the Intelligent and Uniform sampling with the object count for the P01-01 scene changing the total number of sampled frames. 88
5.16 Comparison of frequencies for the Intelligent and Uniform sampling with the object count for each scene at a fixed subsample $\sim$ 1000 frames. 89
5.10 **Qualitative results on P01-01 at 228x128** Visualization of the output of the different models trained on different subsamplings for scene P01-01. The first column represent the real frame while the next ones are respectively: the predicted image, which is the combination of: the static part, the foreground, and the actor part. 92

Acronyms

SfM
: Structure from Motion

pcd
: Pointcloud

MLP
: Multi Layer Perceptron

Introduction

In the last two decades, thanks to the massive technological development that changed our daily lives with things like smartphones or social media, also the world of cameras changed. People wanted to capture unique moments of their lives and share them with their friends. To fulfill this desire *smart devices*[1] and *action cameras*[2] were introduced, tiny robust cameras that could be mounted on the observer's body allowing him to have full use of his arms. This allows the recording of first-person videos, also known as *egocentric* videos, of activities that were difficult if not impossible to capture before.

The abundance of these new types of videos promoted their study, leading to several research aimed to leverage this data. A new branch of Computer Vision specifically for these types of problems has been born, which is referred to as *Egocentric Vision*. Among its tasks, we can mention Action Recognition [1], Object Detection [2] and Segmentation [3], Object Tracking [4], etc. Its main applications found a place in fields such as: (i) *Augmented Reality (AR)* and *Virtual Reality (VR)*, where egocentric data can be used to provide realistic perspectives and interactions as in training simulations or video games; (ii) *Healthcare*, providing assistance and monitoring a patient, helping him with rehabilitation movements, or reminding an old person that she has not turned off the gas stove; (iii) *Security and Surveillance*, where safety can be aided by a virtual assistant that can detect anomalous or suspicious behaviors (some other examples can be found on [5]).

With the emergence of new video data, not only have new research branches been established, but existing fields of study have also been captivated by this new type of data, seeking to address the new challenges it presents. These challenges include *motion blur* caused by the rapid movement of the camera, *limited field of view*, and *occlusion* due to the rich interactions of the user with objects and the proximity of the device to where those interactions happen.

The field of *3D Scene Reconstruction*, also known as *stereophotogrammetry*,

[1]Smart Glasses: Hololens, Aria glasses, Google glass, etc.

[2]Usually wearable devices used to record videos during sports or any other activity that requires mobility or versatility. Some common brands are GoPro, DJI, Insta-360, Sony, etc.

is a branch of photogrammetry that specializes in creating three-dimensional models from image sequences by extracting features from them and then finding correspondences.

Stereophotogrammetry eased the process of acquiring 3D scenes. In fact, 3D reconstructions were, and still are, complex to obtain and manage. The few datasets that contain 3D ground truth have been acquired through static 3D scans of recording locations. This method is unfortunately very expensive both in monetary terms since the hardware scanners are costly, and in time complexity terms. Moreover, those require that a person, possibly specialized or who knows how the scanner works, is physically present on the site. With 3D scene reconstructions based on simple images, we could avoid most of these problems as we would just need a camera that is nowadays pretty cheap and widespread, and a person without any special training that move around the environment as it is. We could even have no access to the physical place as long as we can send there a controlled camera. Some practical examples of how we could exploit the above-mentioned advantages in the future could be space reconstructions [6], obtained via rovers, or medical reconstructions [7], via medical probes. On the other hand, 3D reconstructions could be used in the growing field of augmented reality applications, like in museal applications where architectures and objects can be made available around the world [8], or also landscapes and environments replication that will be used in videogames or metaverse spaces.

However, egocentric videos presented for their nature some additional intrinsic difficulties that made 3D scene reconstruction struggle. First, those are limited by the presence of the actor's body. While the wearer performs any sort of action he/she is usually present in the scene with his arms, occluding part of the scene. This can make the reconstruction process more difficult, as it might not be possible to find any correspondence between the reconstructing images. As a result, the final point cloud that constitutes the 3D reconstruction is an altered representation of the true scene. The reconstruction is soiled by points that represent body parts and occluded parts of the scene are omitted. Another problem is represented by the *interactions* the actor has with his surroundings. If we take as an example a kitchen scenario where a person is cooking, he may be moving or transforming objects in the scene, like slicing some vegetables or changing the location of pans.

This causes these transformations all to be captured in the reconstruction as they coexisted in the same temporal instant, while in reality, they happened at different time instants, resulting in cluttered and misleading reconstructions.

In this thesis, we propose some possible solutions to the issues mentioned above by segmenting dynamic moving objects in 3D and using this information to remove their traces from the reconstructed scene. Possible solutions could be *background subtraction* [9] and *motion segmentation* [10] techniques. However, those are difficult to apply on egocentric videos due to a moving background and motion parallax [11].

Thus, we introduce two different approaches to 3D object segmentation. In both of them, the first step is to obtain a 3D reconstruction of the environment and the motion components in 2D. This is performed using *Structure from Motion* (SfM) [12], a technique used to reconstruct 3D structure from 2D images, and a neural rendering technique, *NeuralDiff* [11], used to decompose the input frames in separate motion components. On top of that, the first approach relies on a *Monocular Depth Estimator* [13], a pre-trained neural network that has learned to predict the depth of each pixel from the camera viewpoint. The depth is then used by the subsequent modules, to project the dynamic pixels from 2D frames to the three-dimensional scene, thanks to geometrical projection formulas. The second one instead exploits the intrinsic knowledge learned from *NeuralDiff*, a combination of three neural networks each one designed to extract a type of motion: static, moving object, and moving actor. On the second approach, we also analyzed how different sampling strategies of the video frames affected the results. As a last step, we modified *NeuralDiff* to better suit our needs by combining two of the three neural layers and thus reducing the computational demand of the overall pipeline.

Results showed that the pipeline using *NeuralDiff* performed well on the egocentric videos. Additionally, our proposed sampling improved over the uniform one.

Reasearch goals and motivations

Our research aims to address the problems introduced by egocentric videos in the 3D reconstruction scenario, like actor's occlusion, motion parallax, motion blur, etc. We worked with one of the largest egocentric datasets, EPIC-KITCHENS [14], and used COLMAP [15], an open-source general-purpose *Structure from Motion* (SfM) pipeline, to reconstruct scenes from egocentric videos. In the second stage, we segmented in two ways objects that move in 3D. The first one is based on a *Monocular Depth Estimator* [16], a pre-trained neural network that has learned to predict the depth of each pixel from the camera. Once the dynamic pixels, detected via *NeuralDiff* [11], are projected in the 3D scene, their distances are taken from the reconstruction points and segmented if closer than a certain threshold. As a matter of fact, we expect the dynamic points to be in the same position, or at least close, to the projected pixels. The second one instead is achieved by a recent neural rendering pipeline: *NeuralDiff*. Neural rendering techniques manage to represent 3D scenes thanks to *Multi Layer Perceptrons (MLPs)*, which learn the mapping of 3D coordinates to their color and opacity as introduced in NeRF [17]. *NeuralDiff* method is designed to divide 2D videos into three separate components, each one represented by a *MLP* namely: *static*, *foreground* and *actor*. We extracted from it the static MLP that can give color and opacity to any 3D point as it belonged to

the static scene. In this way, the opacity was used to segment the point as dynamic or not. If below a certain threshold, the opacity indicates that the point is not visible in the static part, meaning that it belongs to a dynamic object.

In addition, we explored different ways to reduce the computational demands. The first one consists in simplifying the *NeuralDiff* architecture, by merging the two dynamic MLPs in a single one.

The other technique that we explored was to find an intelligent video frame sampling method. We removed any redundant frames while keeping an overall view of the scene and its information. We called this method *Intelligent sampling*.

Results on six kitchens show that Monocular pipeline struggles with egocentric videos while *NeuralDiff* manages to segment dynamic objects in 3D. In addition *Intelligent sampling* reveals to be successful in improving the results of random sampling.

Contributions

In this thesis, we propose two ways to segment dynamic objects in 3D from the challenging egocentric videos. The first step consists in reconstructing the 3D scene. In this stage, we tested the *Structure from Motion* (SfM) [12] algorithm COLMAP [15] on different sampling strategies. We showed that both the number of frames and the resolution revealed to play a key role in a successful reconstruction.

The next step aimed to test methods for segmenting dynamic 3D objects and removing them from the scene. We started by testing the Monocular pipeline by evaluating the precision of the depths estimated from each frame by the *Monocular Depth Estimator* [16] module. Through this pre-trained neural network, we can obtain the depth of each frame. The depth is then used by subsequent modules of the pipeline to obtain the 3D coordinates of the pixels by using geometrical projection formulas. It immediately showed its limitations. The pre-trained neural network seemed to be dependent on the training data as the results were mediocre and in the 12 datasets used as input by [16], no one was egocentric. Also, the monocular-based technique used to segment the point in 3D relied on a threshold distance. This distance is dependent on the scene as each one has a different scale due to the 3D scene reconstruction step, making it difficult to adapt to each new scene.

The second pipeline we proposed is based on the intrinsic knowledge of the *NeuralDiff* [11] component. The method seemed pretty robust as shown both qualitatively and quantitatively. As no 3D ground truth was available for our scenes we used a common strategy used in neural rendering to assess the performances of the network. It consists of projecting from 3D to 2D the learned 3D structure to an image and comparing it with ground truth, the real image and the manually

segmented real image for assessing the segmentation. We tested this pipeline by changing the resolution, the number of frames used, and the sampling procedure of the frames employed.

We compared our new sampling strategy, *Intelligent sampling*, with a standard uniform sampling (e.g., sampling frames uniformly in the video). Results showed that our proposed sampling proved to be successful, capturing the essence of the environment even with a reduced number of frames. Quantitatively, the proposed method is able to achieve 2.9% improvements in terms of PSNR, 1.73% in terms of static PSNR, and 1.78% in terms of mean Average Precision. Qualitatively we provided some proofs that showed objects that were not segmented in the basic uniform sampling were segmented in ours. We also found a correlation between the frequencies of the frames sampled and the frequencies of the actions/objects annotations. This was a good explanation, other than the intuition we gave before, of its functioning, meaning that our sampling focuses on those areas where a lot of actions happen.

As the last step, we proposed *NeuralCleaner*, a simplified version of the *NeuralDiff* pipeline. Since for our purposes, we don't need the distinction between moving objects or actor, we merged the respective *NeuralDiff* layers in a single one. In this way, the training should result to be $\frac{1}{3}$ lighter and the results confirm it, as we obtain a reduced computational time of around $\sim 30\%$.

In summary, our contributions are the following:

- We proposed two methods, one based on *Monocular Depth Estimation* and the other based on *NeuralDiff*, to tackle egocentric video challenges and to segment dynamic objects in 3D.
- We tested *COLMAP*, an SfM technique, on the egocentric dataset EPIC-KITCHENS as the number of frames and their resolution varies.
- We tested a *Monocular pipeline* on egocentric videos for segmenting dynamic objects in 3D.
- We tested how robust the *NeuralDiff* pipeline was to the change in the number of frames and their resolution.
- We proposed a new Intelligent sampling strategy aimed at eliminating redundant frames while keeping important information.
- We simplified *NeuralDiff* in *NeuralCleaner* to better suit our needs and reduce the computational times of $\sim 30\%$.
- Through extensive experiments, we showed that *NeuralDiff* pipeline outperformed the Monocular pipeline.

Part I

Related Works

Chapter 1

Photogrammetry

Photogrammetry involves recording, measuring, and interpreting photographic images and other phenomena to obtain reliable information about physical objects and the environment.

It comprises all techniques concerned with making measurements of real-world object features from images. Its utility ranges from the measuring of coordinates, quantification of distances, heights, areas, and volumes, and preparation of topographic maps, to the generation of digital elevation models and orthophotographs. The functioning relies mostly on optics and projective geometry rules.

In this chapter, we present the basic modelization of cameras and the various assumptions on top of which we will base rules and equations (see Section 1.1). Then we show structure from motion and how to link to images captured from different cameras extracting features in various modalities and then matching with the more appropriate algorithm in Section 1.2. In the last part with Section 1.2.5, we present COLMAP, an open-source SfM that allows us to perform most of the topics we previously explained.

1.1 Photogrammetry Fundamentals

As the first assumption, we can model the camera as a simplified version of itself: the *Pinhole Camera* [18]. As in the first designed cameras (*camera obscura* [19]), in the Pinhole Camera world's light is captured through a pinhole and then projected into the *focal plane*. The main idea is reported in Figure 1.1.

A Pinhole camera is characterized by two collections of parameters:

- **Extrinsic** parameters: they give information on location and rotation in the world.
- **Intrinsic** parameters: gives us internal properties such as: focal length, field

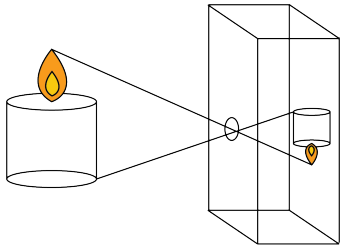

Figure 1.1: Pinhole Camera. A world's object is captured by the camera making light pass through the pinhole and is then projected on the focal plane upside-down.

of view, resolution, etc.

These parameters can be rewritten in their corresponding matrices:

$$\text{Intrinsic} = K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

where f_x, f_y are are the *focal lengths* of the camera in the x and y directions, they are needed to keep the image aspect ratio; c_x, c_y are the coordinates of the *principal point* (the point where the optical axis intersects the image plane).

$$Extrinsic = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ 0_{1\times3} & \mathbf{1}_{1\times1} \end{bmatrix} \tag{1.2}$$

where: $\mathbf{R}_{3\times3}$ is a rotation matrix; $\mathbf{t}_{3\times1}$ is a translation vector.
with $\mathbf{R}$ that can be decomposed in its components:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \quad R_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \quad R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

Extrinsic matrix is also known as the 4x4 transformation matrix that converts points from the world coordinate system to the camera coordinate system.

Exploiting homogeneous coordinates we can rewrite the image-capturing process of a specific camera as the combination of its characteristic matrices:

$$\begin{bmatrix} u \\ v \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{t}_{3\times1} \\ 0_{1\times3} & \mathbf{1}_{1\times1} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1.4}$$

In Figure 1.2 we can see an example of the different reference systems involved in a photogrammetry problem.

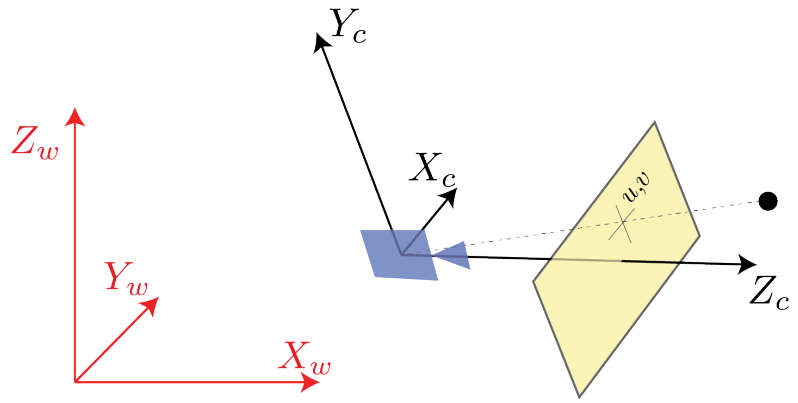

Figure 1.2: Reference systems. An example of different reference systems involved in a photogrammetry problem. In red is reported the world system while in black is the camera one. The yellow plane corresponds to the camera focal plane.

1.2 Structure from Motion: SfM

Structure from motion (SfM) is a technique that estimates the 3D structure of a scene using 2D images. It is used in various applications like 3D scanning, augmented reality, and visual simultaneous localization and mapping.

We can compute SfM in different ways depending on the data and tools at our disposal. Factors such as the **type** and the **number** of cameras can imply using different methods. Also, the **ordering** of the frames may be relevant for a successful reconstruction.

The most basic scenario consists of two images captured from the same camera from two different points of view:

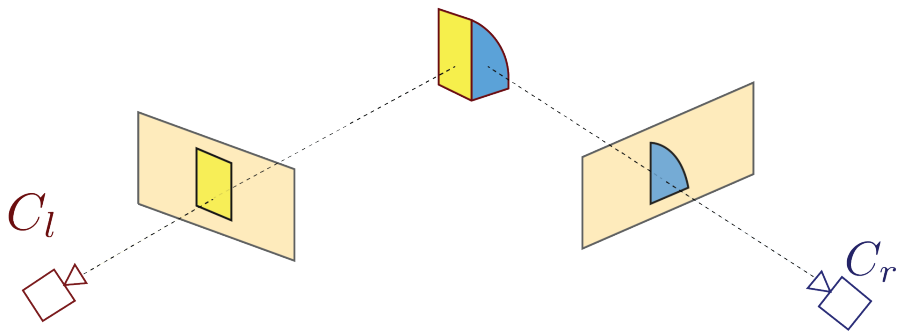

Figure 1.3: Basic SfM Scenario. Two cameras capture the same object from different viewpoints.

In this case, the 3D structure can be recovered *up to scale*, meaning that the relations between the obtained points are faithful to the reality, but it can differ from the real size by a *scale* factor. The scale factor could be retrieved if we know the size of an object in the scene or by having information from other sensors.

In the basic scenario the method consists of the following main steps:

1. **Feature detection** and **Matching**: consists of finding relevant points in each image and finding the corresponding, if they exist, in other images.

2. Estimating **Fundamental matrix**: which describes the geometric relationship between two images of a scene taken from different viewpoints.

3. Estimating **Essential matrix** from fundamental matrix: which represents the intrinsic geometry of the scene, independent of the camera parameters.

4. Estimating **Camera pose** from essential matrix: in this step, the positions of the cameras in the world are extracted.

In the following sections, we describe all these steps.

1.2.1 Feature Detection and Matching

To find corresponding points in two images we need to detect some points of interest in each image. Instead of trying to match each pixel, which would be computationally inefficient and possibly misleading due to color, it has been shown successfully to use *feature points*. Feature points are relevant points that encapsulate the local appearance of its surrounding pixels which is invariant under changes in illumination, translation, scale, and in-plane rotation. Good features are *unique*, *can be easily tracked*, and *can be easily compared*.

A practical example could be to imagine how we humans find correspondences in pictures, looking at Figure 1.4. If we would be asked to find the exact position of the six patches in the underlying image, the job would be easier for patches E and F, being corners they have less possible misleading correspondences, as happens instead of patches A or B.

Figure 1.4: Feature Detection Example [20]. The easier patches to match to the background are the ones with sharp corners.

Similarly, computers extract features. Two of the first algorithms are based on corner detection: **Harris Corner Detector** and **Shi-Tomasi Corner Detector**. Depending on the differences between the two images to be compared, different algorithms have been developed. Here I describe some of them as reported in [20]:

- **SIFT** [21]: Harris corner detector is not good enough when scaling of image changes. Lowe developed a breakthrough method to find scale-invariant features and it is called SIFT. An example of SIFT features is reported in Figure 1.5.
- **SURF** (Speeded-Up Robust Features) [22]: faster SIFT version achieved by using a box filter approximation for the computation of image derivatives instead of Gaussian Filters like in SIFT.
- **FAST Algorithm for corner detection** [23]: All the above feature detection methods are good in some way. However, they are not fast enough to work in real-time applications like SLAM. There comes the FAST algorithm, which is really "FAST".

Figure 1.5: SIFT features position. Example of interest points, we can see that most of them are placed around corners or edges.

- **BRIEF** (Binary Robust Independent Elementary Features) [24]: SIFT uses a feature descriptor with 128 floating point numbers. Consider thousands of such features. It takes lots of memory and more time to match. We can compress it to make it faster. But still, we have to calculate it first. There comes BRIEF which gives the shortcut to finding binary descriptors with less memory, faster matching, and still higher recognition rate.
- **ORB (Oriented FAST and Rotated BRIEF)** [25]: Open source alternative to SIFT and SURF released by OpenCV.

After having found these points of interest, we need to match them with the different pictures. This step is also known as *Feature Matching*. The most basic algorithms are:

- **Brute-Force Matcher** [25]: Given a set of images, for each one, the descriptor of one feature is compared with the one of every other image using a distance metric. The matched feature is the one whose distance is the smallest. It is called Brute-Force because it involves nearly no optimization, taking all possible combinations of images. In Figure 1.6 is reported the output of a Brute-Force Matcher.
- **FLANN** [26]: which stands for *Fast Library for Approximate Nearest Neighbors*. It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features, such as hierarchical k-means clustering. It works faster than BFMatcher for large datasets. FLANN focuses on giving approximate nearest-neighbor search rather than exact matches. Sacrificing some accuracy in favor of speed makes it suitable

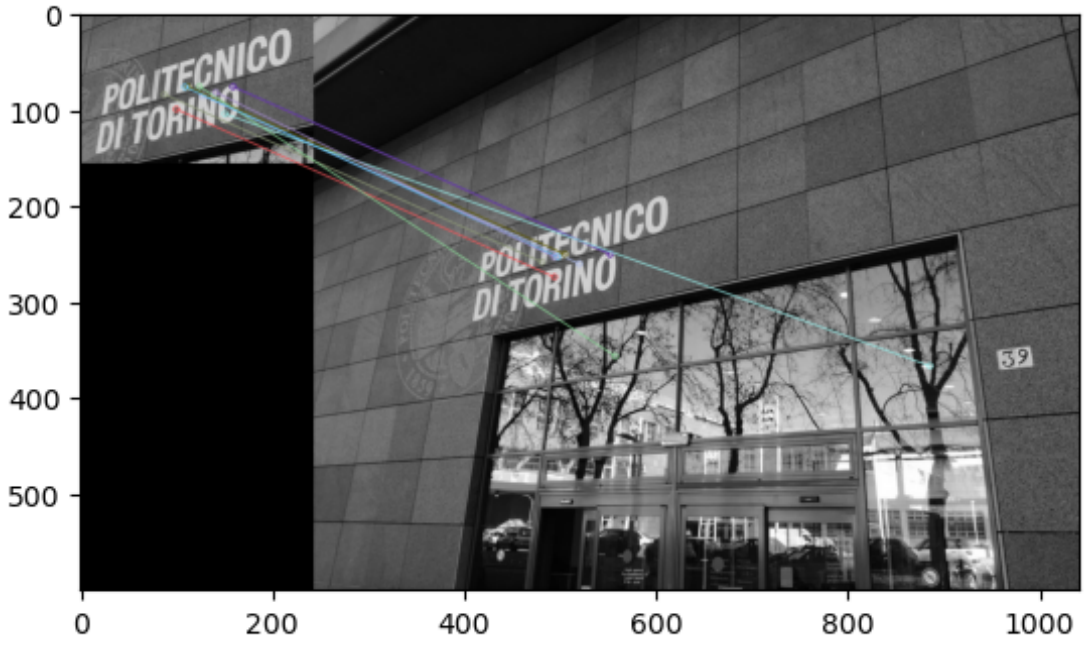

Figure 1.6: Features correspondence computed using BruteForce Matcher on a sample image.

for large-scale datasets. In this way, the feature-matching problem becomes a clustering problem.

1.2.2 Estimating fundamental and essential matrices

The fundamental (F) and the essential (E) matrices allow us to relate the projection of a point located in space from one image to the other. These matrices are based on the so-called *Epipolar Geometry*, which describes the relationship between two images. As we can see from Figure 1.7, given two cameras C_l and C_r the following definitions can be given:

- The **epipole**: which is the point of intersection of the **baseline** (the line that connects the two camera centers C_l and C_r) with the image plane. In Figure 1.7 denoted by e_i.
- An **epipolar plane**: which is any plane containing the baseline.
- An **epipolar line**: which is the intersection of any epipolar plane with the image plane.

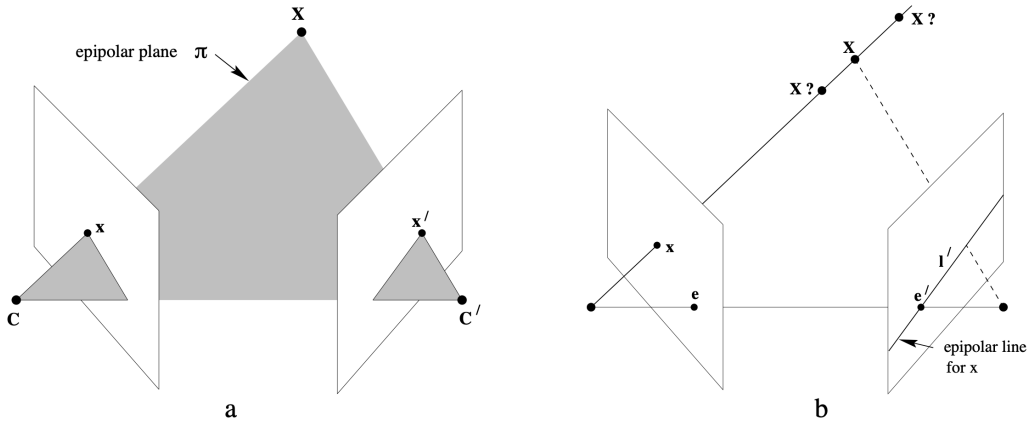

Figure 1.7: Point correspondence geometry. In particular, are reported: the *epipoles* e, e'; the *epipolar plane*, which is any plane containing the line that connects the two camera centers, also known as *baseline*; the *epipolar line* which is the intersection of any epipolar plane with the image plane.

Now that we have a clear understanding of the underlying geometry, we proceed to see the actual derivation of the two matrices.

Consider the following scenario, reported in Figure 1.8: we have a point X and two cameras C_l and C_r. The relative positions are respectively x_l and x_r, while the pixel projections are p_l and p_r. If we consider as reference the left camera, the position of the right one is shifted of a vector $\mathbf{t}$.

We can extract the essential matrix by making the following considerations:

$$x_l \cdot (t \times x_l) = 0 \tag{1.5}$$

but x_l can be written as:

$$x_l = Rx_r + t \tag{1.6}$$

So Equation 1.5 becomes:

$$x_l \cdot (t \times (Rx_r + t)) = x_l \cdot (t \times Rx_r) \stackrel{a}{=} x_l^T \left[t\right]_\times Rx_r = 0 \tag{1.7}$$

where equality (a) is due to the cross-product matrix notation[1]. We call essential matrix the term $\left[t\right]_\times R$, such that:

$$x_l^T \left[t\right]_\times Rx_r = x_l^T Ex_r = 0 \tag{1.8}$$

[1]

$$\mathbf{a} \times \mathbf{b} = \left[\mathbf{a}\right]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

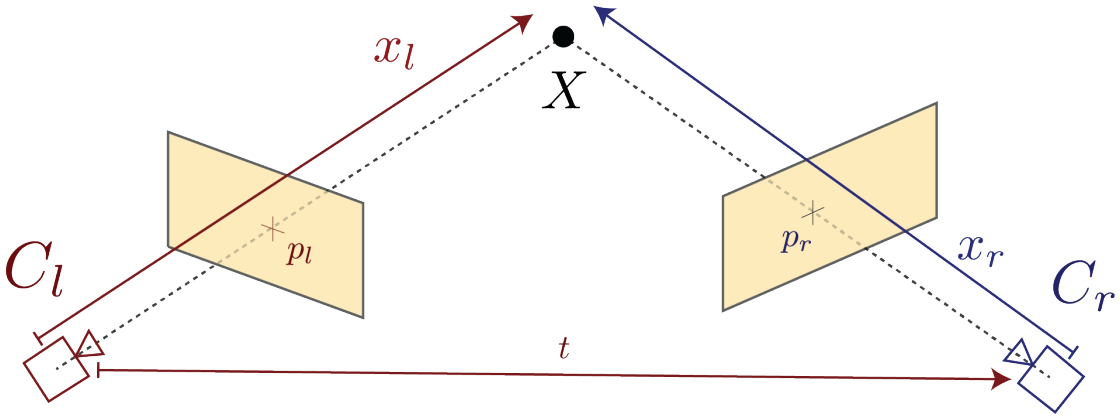

Figure 1.8: A point X viewed from two cameras with x_l being the distance of X from C_l and x_r being the distance of X from C_r. p_r and p_l are instead the projection of X on the respective camera's focal plane.

In a similar way, we can get the fundamental matrix, by replacing the relative positions with the pixel positions. Recalling that the pixel positions are linked to the relative positions by:

$$p_l = \frac{1}{z_l} K_l x_l \quad p_r = \frac{1}{z_r} K_r x_r \tag{1.9}$$

where z_i are the focal distances. We can substitute in Eq. 1.8 and obtain:

$$p_l^T z_l K_l^{-1^T} E K_r^{-1} z_r p_r = 0 \quad z_l, z_r \neq 0 \tag{1.10}$$

Since z_i are constants can be simplified, obtaining:

$$p_l^T K_l^{-1^T} E K_r^{-1} p_r = p_l^T F p_r = 0 \quad z_l, z_r \neq 0 \tag{1.11}$$

where F is the fundamental matrix. We can thus write the relation between the two matrices:

$$E = K_l^T F K_r \tag{1.12}$$

1.2.3 Estimating camera pose from essential matrix

Since $[t]_\times$ is skew-symmetric and R is orthonormal (since it is a rotation matrix), if we know the essential matrix we can decompose it in its components using singular value decomposition, as we can see from Equations 1.13, 1.14, 1.15

$$[t]_\times = UW\Sigma U^T \tag{1.13}$$

$$R = UW^T V^T \tag{1.14}$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.15}$$

1.2.4 Multi-view Structure from Motion

We now explain how we can relate multiple images to recover the structure of a scene. The last stage is called *bundle adjustment* and it is an iterative algorithm used to adjust structure and motion parameters by minimizing a cost function. The possible methods for bundle adjustment are described in [27]:

- **Sequential**: it works by considering additional images at each time, extending in this way the initial reconstruction.
- **Factorization**: it works by computing camera poses and scene geometry using every image measurement at the same time.

Bundle Adjustment is needed since the image measurements are usually noisy. It adjusts the positions and orientations of cameras, as well as the 3D coordinates of points in the scene, to minimize the difference between the observed image features and their predicted locations in the reconstructed 3D model. In simpler terms, it's like fine-tuning a puzzle to make sure all the pieces fit together perfectly, optimizing the alignment of the cameras and the 3D points to improve the accuracy of the reconstruction.

1.2.5 COLMAP

All the sections above focused on a theoretical description of how Structure-from-Motion works. In this section, we illustrate an actual open-source implementation, COLMAP [15].

COLMAP is a software that can reconstruct 3D images using Structure-from-Motion (SfM) and Multi-View Stereo (MVS) techniques. SfM is used to estimate the 3D structure of a scene and camera positions from 2D images taken from different viewpoints, while MVS is focused on creating a detailed 3D model with meshes and textures. COLMAP offers both graphical and command-line interface and can be used on Windows, Linux, and Mac. It is designed for regular desktop computers and servers/clusters and is licensed under the BSD License.

The concepts we explained in the theoretical part are not always applicable in the real world, or their results are not quite satisfying. In literature, a vastity of

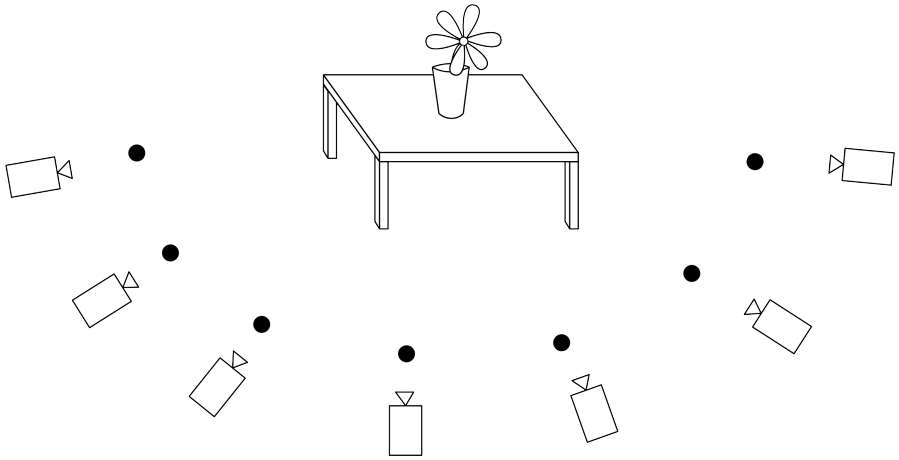

Figure 1.9: Multiple View Scenario. In multiple view scenario multiple cameras are present, each one recording the scene from a different point of view.

Figure 1.10: Building Rome in one day. Result of Rome with 21K registered out of 75K images.

authors tried and succeeded in obtaining refined algorithms for specific scenarios. However, they still lacked a general-purpose method. With COLMAP they managed to compensate for the lack of generalization. The actual implementation and characteristics are explained by the authors in [12, 28] and an example of its usage is shown in Figure 1.10.

Usage

The usage of COLMAP is pretty straightforward. After the creation of a project folder with a *images* directory containing the images we want to process we can simply launch the script reported in Listing (1.1).

Listing 1.1: Automatic COLMAP Reconstruction

```
#!/bin/bash
#The project folder contains a folder "images" with all images.

DATASET_PATH=/path/to/project
colmap automatic_reconstructor \
    --workspace_path $DATASET_PATH \
    --image_path $DATASET_PATH/images \
    --single_camera 1

```

The program offers ample freedom for the single usage of the various steps needed for *SfM* or *MVS*. Here we report the actual pipeline that we used for the extraction of the point clouds for our experiments.

Listing 1.2: COLMAP Single Commands

```
    #!/bin/bash
    DATASET_PATH="/scratch/fborgna/EPIC_Diff/"

    colmap feature_extractor \
        --database_path $DATASET_PATH/database.db \
        --image_path $DATASET_PATH/images \
        --ImageReader.single_camera 1 \

    colmap exhaustive_matcher \
        --database_path $DATASET_PATH/database.db

    mkdir $DATASET_PATH/sparse

    colmap mapper \
        --database_path $DATASET_PATH/database.db \
        --image_path $DATASET_PATH/images \
        --output_path $DATASET_PATH/sparse \
        --camera_model SIMPLE_PINHOLE \

    mkdir $DATASET_PATH/dense

    colmap image_undistorter \
        --image_path $DATASET_PATH/images \
        --input_path $DATASET_PATH/sparse/0 \
        --output_path $DATASET_PATH/dense \
        --output_type COLMAP \
```

```
        --max_image_size 2000

    colmap patch_match_stereo \
        --workspace_path $DATASET_PATH/dense \
        --workspace_format COLMAP \
        --PatchMatchStereo.geom_consistency true

    colmap stereo_fusion \
        --workspace_path $DATASET_PATH/dense \
        --workspace_format COLMAP \
        --input_type geometric \
        --output_path $DATASET_PATH/dense/fused.ply

    colmap poisson_mesher \
        --input_path $DATASET_PATH/dense/fused.ply \
        --output_path $DATASET_PATH/dense/meshed-poisson.ply


```

1.2.6 Monocular Depth Estimation

Until now we have always considered scenarios in which multiple images were available from different points of view. What if we have just one image?

This scenario takes the name of *Monocular Depth Estimation* [16]. As we have previously seen, the projection of a scene on the bi-dimensional image plane of a camera inevitably loses the three-dimensional structure of the scene. In this way, we can no more use optics laws, since we do not have enough information to solve those equations.

Possible solutions include the usage of neural architectures. As a matter of fact, these methods try to learn relations between the possible objects or elements of the image, thus extracting a higher semantic knowledge from the picture. This task is known to be solved particularly well from neural networks, in particular, convolutional or transformer-based networks, trained on large amounts of data, which are nowadays growing in number and quality.

More precisely, the advent of transformers [29] has marked a new state of the art, not only in natural language processing but also in dense prediction. As shown and stated in [16], they "*introduce dense vision transformers, an architecture that leverages vision transformers in place of convolutional networks as a backbone for dense prediction tasks.*[...]*this architecture yields substantial improvements on dense prediction tasks.*[...]*we observe an improvement of up to 28% in relative performance*

when compared to a state-of-the-art fully-convolutional network". The reason for the difference in performance between the two networks lies in their functioning. Although both networks have an encoder-decoder structure, the convolutional layer in one of them limits its performance. Convolutional networks downsample the input image progressively to extract features at multiple scales. However, in dense prediction tasks, it is crucial to maintain feature resolution and granularity. This is because the decoder cannot recover the initial information from compressed features, which could result in poor performance. "*Unlike fully-convolutional networks, the vision transformer backbone foregoes explicit downsampling operations after an initial image embedding has been computed and maintains a representation with constant dimensionality throughout all processing stages*" [16].

The second motivation that made us choose for transformer-based architecture is its versatility and generalizability. In [13] the authors introduce a method that enables mixing multiple datasets during training. In this way, the model will be effective across a variety of scenarios since each dataset is equally varied and captures the diversity of the visual world. Specifically, they experimented with eleven datasets which consist of RGB images with corresponding depth annotation of some form.

In Figure 1.11 we can see some examples of input images with their relative predicted depth map and 3D projection.

Figure 1.11: Top: input images. Middle: Inverse depth maps predicted by the presented approach. Bottom: corresponding point clouds rendered from a novel viewpoint. (Taken from [13])

Chapter 2

Neural Rendering

Neural Rendering refers to a novel set of deep image and video generation techniques that allow for explicit or implicit manipulation of various scene properties like illumination, camera parameters, pose, geometry, appearance, and semantic structure. It combines generative machine learning techniques with a physical understanding of computer graphics to produce controllable and photorealistic outputs.

In this chapter, we will introduce *Neural Radiance Field* (NeRF), and show how it posed an important step in this topic, promoting the birth of new models aimed at different scopes like *NeuralDiff*.

2.1 Previous works

In the next paragraphs, we give a quick review of the work preceding NeRF.

View synthesis and image-based rendering. Photorealistic novel views can be reconstructed if a dense sampling of images has been provided by simple light field sample interpolation techniques [30, 31]. If sparser images are available, some methods have been introduced that rely on extracting traditional geometry and appearance representations from observed images. A popular class of approaches uses mesh-based representations of scenes with diffuse [32] or view-dependent [33] appearance. Also, there are differentiable rasterizers [34, 35] that can directly optimize mesh reconstruction to reproduce a set of input images using gradient descent. However, due to local minima or poor conditioning of the loss landscape, the optimization is usually difficult.

High-quality photorealistic view synthesis is also performed by volumetric representations, from a set of given RGB images. Volumetric methods can represent complex shapes and materials, well suited for gradient-based optimization. First

works [36] used observed images to directly predict color voxel grids. Later [37, 38, 39] used deep networks to color a sampled volumetric region from some given images.

However, these methods, being based on discrete sampling of the voxel grid, are restricted to low resolution due to time and computational power. With neural rendering instead a *continous* representation was proposed, reducing the memory requirements and producing higher-quality renderings.

Background subtraction Background subtraction techniques have been used to detect moving objects in videos [9]. The simplest way to obtain the background would be to capture a background image that does not contain any foreground object. Unfortunately in some scenes, it is not possible and it could be changed under critical situations like illumination changes, objects appearing and disappearing from the scene. Some expedients have been introduced that try to predict the background via a background initialization step, which bases its guess on the first few frames of the video [9]. However, egocentric videos still contain too many challenging problems (light change, multiple different objects, moving camera, etc.) that do not allow background subtraction to be a viable solution.

Motion segmentation Motion segmentation consists of decomposing a video into individually moving objects [10]. Amongst others, it has applications in robotics, traffic monitoring, sports analysis, inspection, video surveillance, and compression. However, these techniques usually rely on optical flow, which can be subject to some ambiguities like, in our case, motion parallax. Even occlusion plays an important role. Motion segmentation struggles for example when an object moves in front of or behind other objects in the scene, leading to ambiguity in the flow field. Also, many methods fail when a dynamic object temporarily remains static.

Discovering and segmenting objects in videos Discovering and segmenting objects in videos is related to background subtraction and motion segmentation. For instance, moving objects can be segmented from the background by using a probabilistic model that acts on optical flow [40]. In [41], pixel trajectories and spectral clustering are combined to produce motion segments. Some recent works revisit classical motion segmentation techniques from a data-driven perspective [42], *e.g.* using physical motion cues to learn 3D representations or learning a scene representation using neural rendering (see Section 2.2).

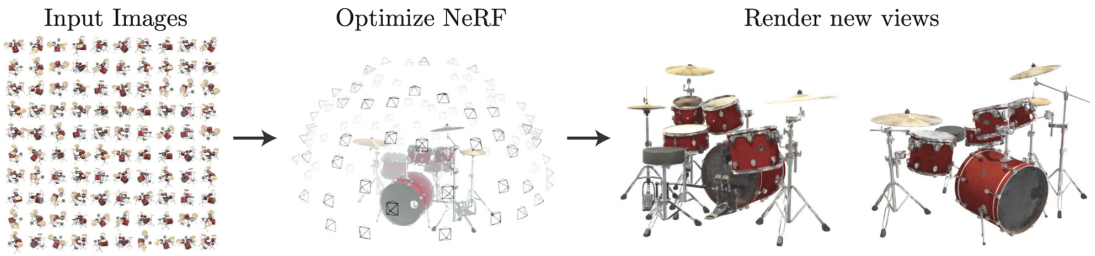

Figure 2.1: NeRF. Optimization of a continuous 5D neural radiance field representation (volume density and view-dependent color at any continuous location) of a scene from a set of input images. The 2D novel views are obtained thanks to classic volume rendering techniques. Here in this example, given 100 images acquired from different viewpoints, they sample two novel views.

2.2 NeRF: Representing scenes as neural radiance fields for view synthesis

With NeRF [17] the authors introduced a new state-of-the-art model for synthesizing novel views of complex scenes by just using a set of sparse images and their relative positions. Figure 2.1 is reported as an example of the main steps from the original paper.

2.2.1 NeRF

A continuous 3D scene is represented as a 5D vector-valued function whose input is a 3D coordinate $\mathbf{x} = (x, y, z)$ and 2D viewing direction $\mathbf{d} = (\theta, \phi)$, and whose output is an emitted color $\mathbf{c} = (r, g, b)$ and volume density σ. In practice, the scene is represented by an MLP network $F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$, where Θ are the weights of the network. This was given the name of *Neural Radiance Field (NeRF)*.

The representation is encouraged to be multiview consistent by restricting the network to predict the volume density σ as a function of just the location $\mathbf{x}$, while the color $\mathbf{c}$ is a function of both the input, location and direction. To do this the MLP F_Θ first processes the input $\mathbf{x}$ with 8 fully-connected layers (using ReLU activation functions and 256 channels per layer), and outputs σ and a 256-dimensional feature vector. The scheme is summed up in Figure 2.2.

The effects of the direction in input can be seen in Figure 2.3.

Volume rendering with radiance fields

The implicit representation of the scene relies on the volume densities and on the color of every point in that scene. The color of any ray passing through the scene is

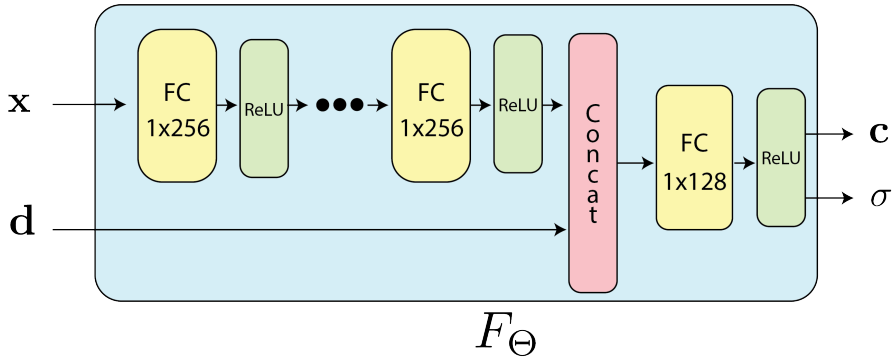

Figure 2.2: F_Θ **Scheme.** The input position **x** pass through 8 Fully connected (FC) layers of 256-channels. Each FC layer is followed by a ReLU activation function. This intermediate result is then concatenated with the input direction (**d**) and fed to one last FC with 128 channels that feed its output to a ReLU function. The output of the ReLU is the color **c** and the volume density (σ).

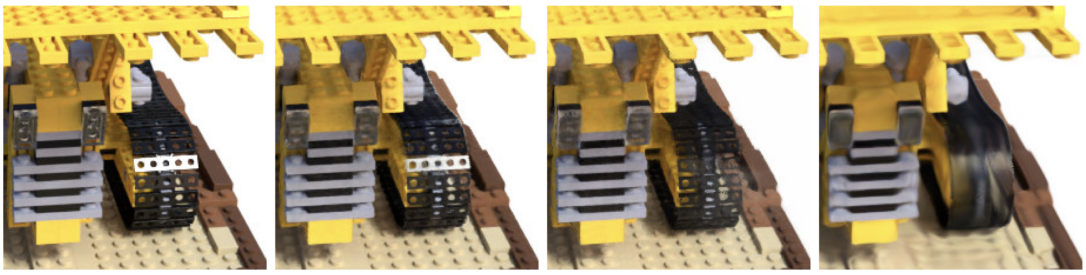

Figure 2.3: Here we reported the results obtained with different strategies, as written underneath each image. In particular, removing view dependence prevents the model from recreating the specular reflection on the bulldozer tread. Removing the Positional encoding instead, we obtain a blurred image, meaning that high frequencies are not captured nor represented.

rendered using principles from classical volume rendering [43]. The volume density $\sigma(\mathbf{x})$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location **x**. While the expected color C(**r**) of camera ray **r**(t) = **o**+ t**d** with near and far bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r(t)},\mathbf{d})\, dt \tag{2.1}$$

where $T(t)$ denotes the accumulated transmittance along the ray from t_n to t, i.e: the probability that the ray travels from t_n to t without hitting any other particle.

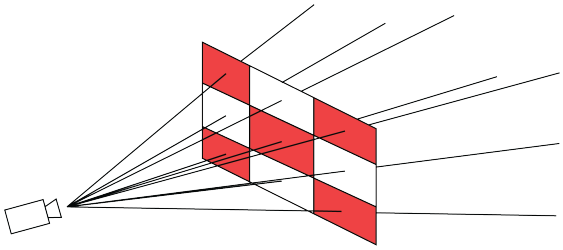

Figure 2.4: Example of rays passing through an image plane of size 3x3 pixels.

Namely:

$$T(t) = exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds) \tag{2.2}$$

To obtain a new view in our neural radiance field, we should estimate the $C(\mathbf{r})$ function for each ray passing through each pixel of the focal plane (see Figure 2.4). This integral is approximated using a numerical method known as *quadrature*. Typically deterministic quadrature is used for rendering discretized voxel grids, but in our case, it would limit our model's resolution since the network would only be queried at a fixed discrete set of locations. To solve this problem a *stratified* sampling approach has been used, where each interval $[t_n, t_f]$ has been partitioned into N evenly-spaced bins, from which a random sample is then uniformly extracted. Namely, as in the following:

$$t_i \sim \mathcal{U}[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)] \tag{2.3}$$

In this way, even if we are using a discrete set of samples, using stratified sampling allows us to represent a continuous scene representation because the network is evaluated at continuous positions during the training phase. Following the quadrature rule discussed in [44] they used the samples to estimate $C(\mathbf{r})$:

$$\widehat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))\mathbf{c}_i, \quad where \quad T_i = exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j) \tag{2.4}$$

where $\delta_i = t_{(i+1)}$ is the distance between adjacent samples. It can be noticed that the function that approximates $C(\mathbf{r})$ is differentiable and can be reduced to traditional *alpha compositing*[1] with alpha values being $\alpha_i = 1 - exp(-\sigma_i\delta_i)$.

[1] *Alpha compositing* is a digital imaging technique used to combine multiple layers of images or graphics with transparency, known as alpha channels.

Training procedure

To optimize a scene RGB frames are required with the corresponding camera poses and intrinsic parameters (for synthetic data these are easily retrievable from the scene model; while for real images COLMAP was used to extract them). At each iteration a batch of rays from the set of all pixels of all images of the dataset is extracted and following the sampling procedure previously described the actual color is predicted for both the "coarse" and the "fine" model. All the previous steps are differentiable such that we can use gradient descent to optimize the model by minimizing the loss function that is computed as the total squared error between the rendered and true pixel colors for the two models:

$$\mathcal{L} = \sum_{r \in \mathcal{R}} [\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2] \tag{2.5}$$

where $\mathcal{R}$ is the set of rays of each batch, $C(\mathbf{r})$ is the RGB color ground truth and $\hat{C}_c(\mathbf{r})$, $\hat{C}_f(\mathbf{r})$ are the predicted color for the "coarse" and the "fine" model for ray $\mathbf{r}$.

Optimizing a neural radiance field

In the previous sections, the core components of a Neural Radiance Field were presented, yet, these parts alone are not able to achieve state-of-the-art results. To improve the quality of the representation two improvements were found successful:

- **Positional encoding** of the input coordinates, to encourage the representation of high-frequencies.
- **Hierarchical sampling** procedure that allows for efficiently sample high-frequency representation.

Positional encoding After having found that the model F_Θ performed poorly operating solely on $xyz\theta\phi$ input, in accordance to the work by Rahaman *et al.* [45] that states that deep networks are biased towards learning low-frequency functions; they encode the inputs to a higher dimensional space using high-frequency functions.

The new model F_Θ can thus be represented as a combination of functions $F_\Theta = F'_\Theta \circ \gamma$, where γ is a function from $\mathbb{R}$ to a higher dimensional space $\mathbb{R}^{2L}$, and F'_Θ is still the basic block introduced in the previous sections. Formally the function used for the encoding part is:

$$\gamma(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^L - 1\pi p), cos(2^L - 1\pi p)) \tag{2.6}$$

In particular $\gamma(\cdot)$ is applied separately to each component of $\mathbf{x}$ and $\mathbf{d}$ after these values are normalized to lie in $[-1,1]$. Reporting the paper results, they found out that good values of L were: 10 for $\gamma(\mathbf{x})$; and 4 for $\gamma(\mathbf{d})$.

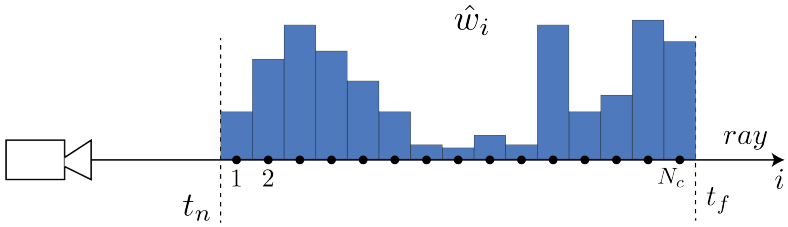

Figure 2.5: Probability Density Function of normalized coarse weights $\hat{w}_i$ along a ray with N_c samples.

Hierarchical sampling Stratified sampling allows to cover continuous regions but however is still inefficient: free space and occluded regions that do not contribute to the rendered image are still sampled repeatedly. To solve this problem the authors proposed a *hierarchical* representation which increased rendering efficiency by distributing samples proportionally to their expected effect on the final rendering.

This solution consists of simultaneously optimizing two networks: one "coarse" and one "fine". The first step expects to sample a set of N_c points using stratified sampling and feed those to the coarse model. Once this first partial result has been obtained a more informed sampling of points is produced. The coarse sampling allows us to get a rough idea of which parts of the volume are the most relevant. To do this they rewrite the alpha composited color from the coarse model $\hat{C}_c(\mathbf{r})$ in Eq. 2.4 as a weighted sum of all sampled colors c_i along the ray:

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, \qquad w_i = T_i(1 - exp(-\sigma_i \delta_i)) \tag{2.7}$$

By normalizing the weights as $\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_c} w_j}$ we can produce a piecewise-constant probability density function along the ray as seen in Figure 2.5. The next N_f samples are extracted from this distribution and then, combined with the previous N_c samples, fed to the "fine" network. The final rendered color $\hat{C}_f(\mathbf{r})$ is obtained using Eq. 2.4 with $N_c + N_f$. This procedure revealed successful in obtaining more samples from regions we expect to contain visible content.

2.2.2 Results

A qualitative idea of the potentiality of NeRF is reported in Figure 2.6 and 2.7. Specifically, in the first Figure is reported the comparison of NeRF, Local Light Field Fusion LLFF [46], and Scene Representation Network SRN [47] on the synthetic data introduced in [17]. We can see that NeRF is able to retrieve even the smaller details and no artifacts that are present in other methods are produced. On the

other hand, in the second figure, the algorithms are compared on real images and NeRF shows better performance with respect to other methods.

Figure 2.6: Comparison of test images from the newly introduced synthetic dataset. The algorithms compared are NeRF, Local Light Field Fusion LLFF [46], and Scene Representation Network SRN [47]. The NeRF method can recover fine details in both geometry and appearance. LLFF exhibits some artifacts on the microphone and some ghosting artifacts in the other scenes. SRN produces distorted and blurry rendering for every scene. Neural Volumes struggle to capture details we can see from the ship reconstruction.

Figure 2.7: Comparison on the test set of the real images. As expected LLFF is performing pretty well being projected for this specific use case (forward-facing captures of real scenes). However, NeRF can represent fine geometry more consistently across rendered views than LLFF as we can see in Fern's and T-rex. NeRF is also able to reproduce partially occluded scenes as in the second row. SRN instead completely fails to represent any high-frequency content.

2.3 NeuralDiff: segmenting 3D objects that move in egocentric videos

NeuralDiff is a neural radiance field adapted to distinguish three different parts of egocentric videos:1) *static*, that is everything that does not move, 2) *foreground*, which is everything that at some point of the video moves, and 3) *actor*,which comprehends the body parts of the person who is wearing the camera.

Still, this is not an easy task. Motion in egocentric vision is a complex attribute to identify. We need to distinguish what is independent of the camera motion while dealing with the camera's large viewpoint change and parallax that generate a large apparent motion.

In a static camera scenario, the problem of separating the foreground from the static background would be easily solved by recurring to background subtraction techniques, but the parallax effect of a moving camera makes this technique useless. As an example provided in the paper, we can think of an egocentric video of a person cooking: the actor behaves in the scene by moving (and transforming) objects. However, ego-motion is the dominant effect since objects move only sporadically, and in a way that is hardly distinguishable from the much larger apparent motion induced by the viewpoint change, making it very difficult to segment dynamic objects automatically.

Even motion segmentation techniques struggle to separate a scene into different motion components, since they require correspondences, they reason locally, across a handful of frames and usually avoid explicit 3D reasoning, making it difficult to treat egocentric videos with many small objects that move only occasionally throughout a long sequence [10].

With this work, the authors take inspiration from neural rendering techniques [17] to create a motion analysis tool to obtain their goal. In particular, they leverage the ability to reconstruct accurately 3D scenes to recover the background and then build on top of it the dynamic parts. 3D objects that are manipulated in the video also present a significant structure. They usually move in "bursts", changing their state when they are involved in an interaction, otherwise staying rigidly attached to the background. Exploiting this property they extend the neural render to reconstruct the moving object's appearance using a slowly-varying time encoding. The last part that shows unique properties in the scene is the actor due to his continuous movement while occluding the scene and with a motion linked to the camera. In Figure 2.8 is reported a general overview of NeuralDiff capabilities.

Neural rendering for dynamic videos. The spread of neural rendering with NeRF (Section 2.2) paved the way for new research also in dynamic scenarios (see NeRF-W [48]). Another direction tried focusing on modeling dynamic scenes

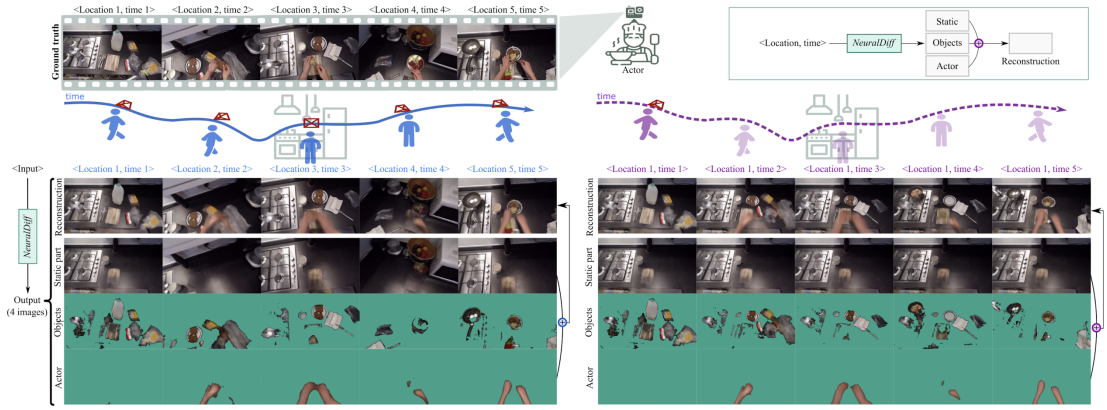

Figure 2.8: Given an egocentric video with camera reconstruction, NeuralDiff, a neural architecture, learns how to decompose each frame into a static background and a dynamic foreground, which includes every object that sooner or later will move and the actor's body parts. Each of these streams is learned by exploiting the characteristics of the scene that is going to be captured. Being a neural radiance field, NeuralDiff is also capable of rendering images from novel viewpoints as can be seen in the bottom right part of the scene.

mostly with monocular videos as input [49, 50]. Most of these approaches use a canonical model in conjunction with a deformation network, or warp space [50], starting from a canonical volume. Closer to the work presented is [49], where a static NeRF model is combined with a dynamic one.

Nonetheless none of the previous managed to segment 3D objects in such long and challenging videos. We next describe how NeuralDiff succeeded in it.

2.3.1 NeuralDiff

Static components

As reported in Section 2.2, a video $\mathbf{x}$ is a collection $(x_t)_{t\in[0,\dots,T-1]}$ of T RGB frames $x_t \in \mathbb{R}^{3\times H\times W}$. Each of these frames can be seen as a function $x_t = h(B, F_t, g_t)$, where B is the static background, F_t is the variable Foreground, and g_t is the moving camera. The motion and parameters of the camera are assumed to be known, usually being extracted via a Structure from Motion (SfM) algorithm such as COLMAP [15] which is explained in detail in Section 1.2.5. The background and foreground layers include the shape and reflectance of the 3D surfaces in the scene as well as the illumination.

Instead of trying to invert the function h to get B and F_t, which is known as *inverse rendering*, neural rendering directly learns h using a neural network f,

$h(B, F_t, g_t) = f(g_t, t)$, provided the time and the camera viewpoint for the frame x_t. Viewpoint g and time t can be factorized by a careful design of the function f, thus f can be used to generalize new unobserved viewpoints. In the basic NeRF implementation, the main assumption is that the scene is static, meaning that F_t is empty and f can be rewritten as $f(g_t)$. The color of each frame is then obtained by a volumetric sampling process that simulates ray casting. More in detail the pixel color $x_{ut} \in \mathbb{R}^3$ of pixel $u \in \Omega = \{0, ..., H-1\} \times \{0, ..., W-1\}$ is obtained by averaging the color of the 3D points along the ray $g_t r_k$ weighted by the probability that a photon emanates from the point and reaches the camera.

A neural network, $(\sigma_k^b, c_k^b) = MLP^b(g_t r_k, d_t)$ retrieve the density $\sigma_k^b \in \mathbb{R}_+$ and the color $c_k^b \in \mathbb{R}^3$ of each point $g_t r_k$, where the superscript b refers to the fact that we are dealing with the static background and d_t is the unit-norm viewing direction.

A photon while travelling along the segment (r_k, r_{k+1}) is transmitted with probability $T_k^b = e^{-\delta_k \sigma_k^b}$ where the quantity $\delta_k = |r_{k+1} - r_k|$ is the length of the segment. This definition allows us to calculate the probability of transmission across several segments as the product of the individual transmission probabilities. The color of pixel u can thus be written as:

$$x_{ut} = f_u(g_t) = \sum_{k=0}^{M} v_k (1 - T_k^b) c_k^b, \quad v_k = \prod_{q=0}^{k-1} T_q^b \tag{2.8}$$

where $f_u(g_t)$ is the function that maps the viewpoint to the single pixel u. The model is trained by minimizing the reconstruction error $\|x - f(g_t)\|$.

Dynamic components

To reconstruct egocentric videos, we deal with moving objects, so we can not neglect the foreground F_t. To capture this layer they propose to build on top of the background MLP^b a foreground-specific MLP^f, $(\sigma_k^f, c_k^f, \beta_k^f) = MLP^f(g_t r_k, z_t^f)$. Its outputs are a "foreground" occupancy σ^f and color c^f. It also predicts β_k^f, an uncertainty of the color associated to each 3D point along the ray r_k and whose role is discussed in the next paragraphs. The variable z_t^f is introduced to capture the properties of the foreground that change over time.

The color x_{ut} of a pixel u is obtained by the composition of both background and foreground, so $\mathcal{S} = \{b, f\}$:

$$x_{ut} = f_u(g_t, z_t) = \sum_{k=0}^{M} v_k (\sum_{p \in \mathcal{S}} w^p(T_k) c_k^p) \quad where \quad v_k = \prod_{q=0}^{k-1} \prod_{p \in \mathcal{S}} T_q^p \tag{2.9}$$

The factor v_k requires a photon to be transmitted from the camera to point r_k through different materials. The weights $w^p(T_k)$ mix the colors based on point

densities. As done in NeRF-W [51]:

$$w^p(T_k) = 1 - T_k^p \in [0,1] \tag{2.10}$$

Each stream is designed differently, to include inductive biases that match the statistics of each layer (background, foreground, actor). The resulting analysis-via-synthesis method shows that neural rendering techniques are also useful for analysis, and not just synthesis.

Smooth dynamics The model can now be optimized by minimizing the loss across all input frames:

$$\min_{f,z_1,...z_T} \frac{1}{T|\Omega|} \sum_{t=1}^{T} \sum_{u\in\Omega} \|x_t - f(g_t, z_t)\|^2 \tag{2.11}$$

However, there is a problem with the characteristics of the foreground, because, even if dynamic, it does not change at every frame most objects spend most of their time rigidly attached to the background. This makes the dependence on independent frame-specific codes z_t almost useless. They came up with the idea of replacing it with a low-rank expansion of the trajectory of sates, taking $z_t = B(t)\Gamma$ where $B(t) \in \mathbb{R}^p$ is a fixed basis and the motion $\Gamma \in \mathbb{R}^{P\times D}$ are coefficients such that $P << T$. In particular $B(t) = [1, t, sin2\pi t, cos2\pi t, sin4\pi t, cos4\pi t, ...]$

Improved geometry: capturing the actor The foreground layer can be divided again into objects manipulated by the actor, Which can move sporadically, and the actor, which instead moves continually. To detect the actor a third MLP is integrated into the previous model. The actor's MLP is similar to the foreground MLP: $(\sigma_k^a, c_k^a.\beta_k^a) = MLP^a(r_k, z_t^a)$, where the a stays for actor. The main difference is that this time the 3D point r_k is expressed with respect to the camera, and not to the world ($g_t r_k$). This follows the physical properties of the recording stage, the camera is attached to the actor's head, making his body parts almost always present in front of the camera therefore it shows a reduced variability in the reference frame of the camera. On the contrary, the background and foreground are invariant to the world reference system.

Improved color mixing A principled mixing model is proposed to replace Equation 2.10. It is obtained by dividing the segment δ_k into Pn sub-segments, alternating between the P different materials ($P = |\mathcal{S}|$, $e.g. P = 3$ if all three layers are considered). They show that the probability that a photon is absorbed in a subsegment of material p is given by:

$$w^p(T_k) = \frac{\sigma_k^p}{\sum_{q=1}^{P} \sigma_k^q}(1 - \prod_{q=1}^{P} T_k^q) \tag{2.12}$$

where the first term represents the probability that a given material p is responsible for the absorption. The latter one instead is the probability that the photon is absorbed by all the materials involved.

Uncertainty and regularization

Uncertainty. Here we clarify the role of the previously announced β_k^p variable. It is predicted from each MLPs ($\beta_k^b = 0$ for the background) and represents the uncertainty of the color associated with each 3D point along the ray r_k for each layer p as pseudo-standard deviations (StDs). As reported in [48], the StD of a rendered color x_{ut} is given as the sum of all the StDs for each material: $\beta_{ut} = \sum_p \beta_{ut}^p$ where β_{ut}^p is obtained from Equation 4.4 by replacing c_k^p with β_k^p. Now we can introduce a probabilistic loss:

$$\mathcal{L}_{prob}(f, z_t | x_t, g_t, u) = \frac{\|x_{ut} - f_u(g_t, z_t)\|^2}{2\beta_{ut}^2} \tag{2.13}$$

Sparsity. The occupancy of the foreground and actor components is further penalized by using:

$$\mathcal{L}_{sparse}(f, z_t | x_t, g_t, u) = \sum_{p=1}^{P} \sum_{k=0}^{M} \sigma_k^p \tag{2.14}$$

Training loss. Finally, the loss used for training the model is:

$$\mathcal{L} = \mathcal{L}_{prob} + \lambda \mathcal{L}_{sparse} \tag{2.15}$$

where $\lambda > 0$ is a weight set to 0.01.

2.3.2 EPIC-Diff benchmark

The goal of this paper was to identify any "detachable" object, namely a object that moves independently from the camera. An extension of EPIC-KITCHENS [14] was introduced to give an evaluation to their method.

Data selection. 10 video sequences, or also called *scene*, were selected from EPIC-Kitchen, each lasting 14 minutes on average. Then 1000 frames were sampled from each and fed to COLMAP [15] to obtain camera reconstructions. The scenes followed these constraints. 1)The videos should contain different viewpoints and multiple manipulated objects. 2) COLMAP should reconstruct the sequence with at least 600 frames. In the end, they obtain 10 sequences with an average of 900 frames.

Figure 2.9: Examples of frames with their corresponding manually segmented binary pixel-wise masks.

Data annotation. Being an unsupervised algorithm, the only data annotations that were collected for testing and validation. They uniformly hold out 56 frames on average for validation (for setting parameters) and for testing. The latter were manually annotated with segmentation pixel-wise binary masks to assess static/dynamic components. The test frames are not used for training such that they can be used also to evaluate novel-view synthesis. In total, they obtained 560 manual image-level segmentation masks. An example can be see in Figure 2.9

Evaluation. The task evaluated is background subtraction. To accomplish this they used standard segmentation metrics: each frame is decomposed in its pixels and a mask is extracted from the predicted frame. Then it is compared with its ground-truth calculating average precision (AP). Each AP is then averaged with every frame and scene to obtain mean average precision (mAP).

For novel view synthesis instead, PSNR is used. Specifically, they provided, using the ground-truth masks, the PSNR of the static and the moving parts.

Results. The experiments are based on a PyTorch implementation of the model that has been published on `https://github.com/dichotomies/NeuralDiff`, more details can be found in the paper [11].

The baselines compared are:

(1)**NeRF** [17] which uses a single stream to predict static scenes.

(2)**NeRF-BF** trains two NeRF models in parallel, one for the Background (B) and the other for the foreground (F). The latter stream is conditioned on time by passing a positional-encoded version of the time variable, that in this case corresponds with the frame number. This differs from NeuralDiff time encoding for Equation 2.11.

(3)**NeRF-W** [48] also contains two interlinked background and foreground streams. It was designed to deal with image collection, so it struggles to adapt to videos. Some adjustments were made to adapt it to this task, more details are reported in the paper.

Qualitative Results. We report just some qualitative results to grasp the

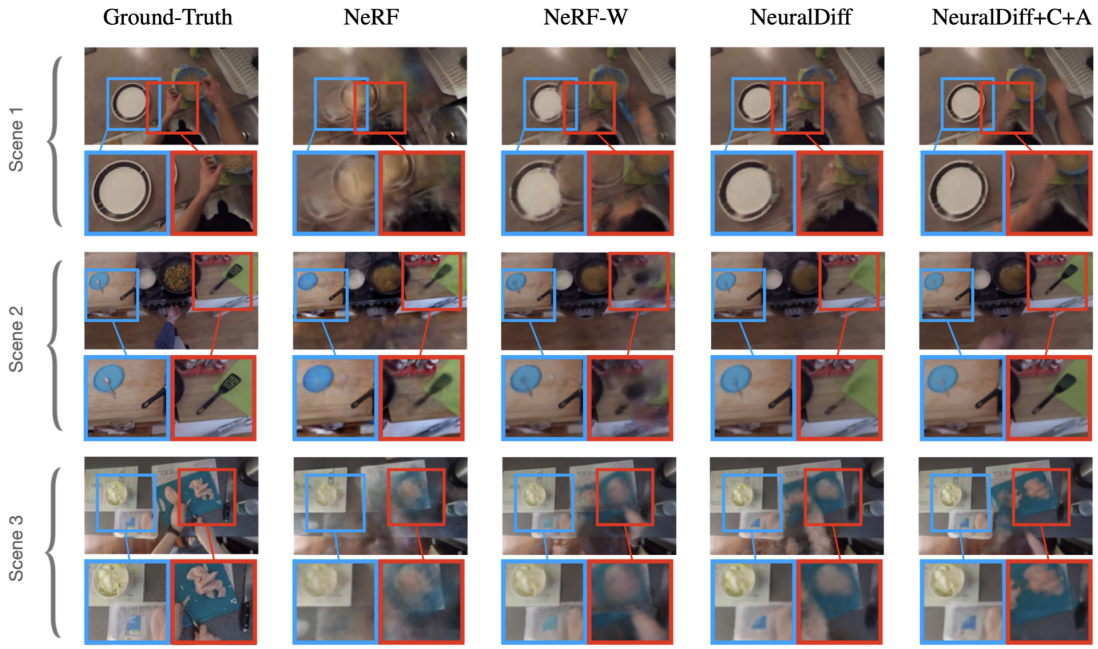

Figure 2.10: Three Scenes reconstruction from NeRF, NeRF-W and NeuralDiff, NeuralDiff+C+A. It can be seen that NeuralDiff produces less ghosting artifacts, captures most moving objects and shows more details, especially the *upgraded* version.

effective results obtained from their method. In Figure 2.10 the output of the compared methods is reported for three different scenes. It can be seen that NeuralDiff produces less ghosting artifacts, captures most moving objects and shows more details, especially the *upgraded* version. In particular: NeRF struggles with the dynamic components and creates blurry reconstructions; NeRF-W obtains sharper static images but still struggles with dynamic regions; NeuralDiff produces sharper results but the *upgraded* version captures more details, such as the arms in the first scene or the spoon in the second. Another qualitative result is shown in Figure 2.11. In the best segmentation case for NeuralDiff+A, NeRF-WW creates noisy results, where the plate, the pasta colander, and the actor's body are barely captured. NeuralDiff instead manages to capture all moving objects and more body parts but misclassifies the floor as foreground. NeuralDiff-A better estimates the shape of the actor's body (second and third row) and correctly recognize the floor as static.

For the failure case instead, NeuralDiff still beats NeRF-W. In detail, NeRF-W fails to recognize any foreground object. NeuralDiff performs a little better by discriminating some foreground objects but still including some of the static

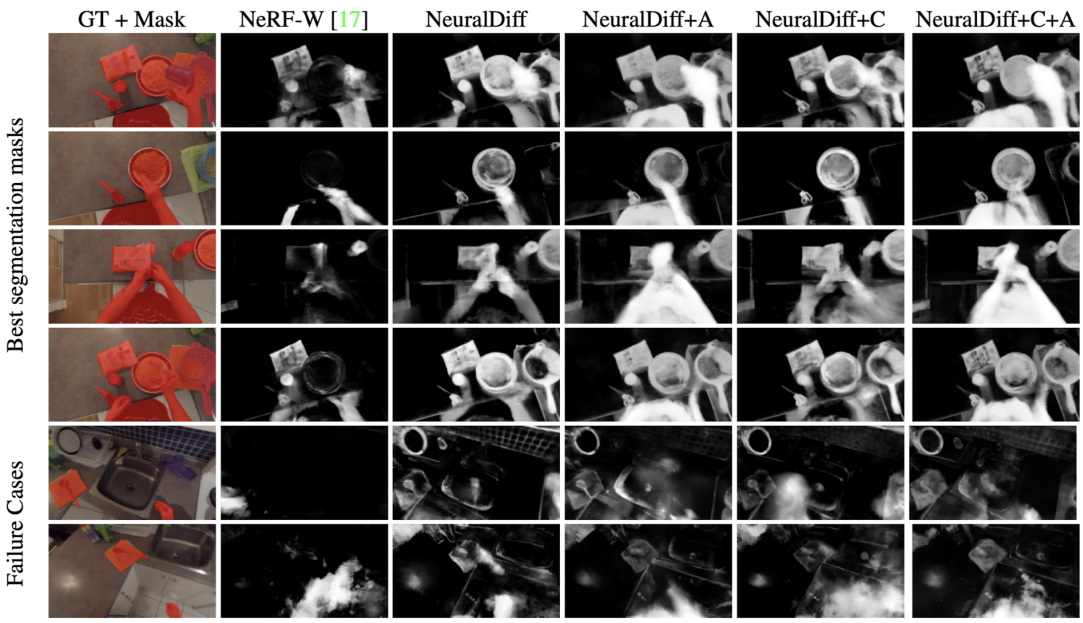

Figure 2.11: Segmentation Masks. Here the masks for which NeuralDiff scored best (top 4 rows) and worst (last 2 rows) are reported.

background as foreground (e.g. last row, part of the sink is reported as foreground).

Chapter 3

Datasets

In this chapter, we are going to present some of the currently largest egocentric datasets. Following a chronological order we started by introducing EPIC-KITCHENS in Section 3.1 upon which the remaining are built. EPIC-KITCHENS 100 in Section 3.2 is an enlargement of its data. EPIC-Fields in Section 3.3 extends its frames with 3D camera information while at Section 3.4 VISOR introduces pixel segmentation annotations.

3.1 EPIC-KITCHENS

EPIC-KITCHENS [14] is the largest and most varied dataset in egocentric vision up to our knowledge. It contains 55 hours of annotated video data recorded by a head-mounted camera of nonscripted actions, meaning that the actors were not following any *scripted* actions (we will see this in more detail later).

3.1.1 Motivation

EPIC-KITCHENS was born to fill the gap in the scarcity of annotated video datasets. As a leading comparison, at the time of writing significant progress has been seen in many domains such as image classification [52], object detection [53], captioning [54] and visual question answering [55]; due to the advances in deep learning but mainly due to the availability of large-scale image benchmarks such as PASCAL VOC [56], ImageNet [57], Microsoft COCO [58], ADE20K [59]. In the same way, the authors thought that introducing a large-scale video dataset could contribute to the development of video domains.

Some video datasets were already available for action classification [60, 61, 62, 63, 64] but, a part from [63], these all contain very short videos, focusing on just a single action. A solution to this problem was given by Charades [65] where 10k

Dataset	Ego?	Non-Scripted?	Native Env?	Year	Frames	Sequences	Action Segments	Action Classes	Object BBs	Object Classes	Participants	No. Env.s
EPIC-KITCHENS	✓	✓	✓	2018	11.5M	432	39,596	149*	454,255	323	32	32
EGTEA Gaze+ [16]	✓	×	×	2018	2.4M	86	10,325	106	0	0	32	1
Charades-ego [41]	70%✓	×	✓	2018	2.3M	2,751	30,516	157	0	38	71	N/A
BEOID [6]	✓	×	×	2014	0.1M	58	742	34	0	0	5	1
GTEA Gaze+ [13]	✓	×	×	2012	0.4M	35	3,371	42	0	0	13	1
ADL [36]	✓	×	✓	2012	1.0M	20	436	32	137,780	42	20	20
CMU [8]	✓	×	×	2009	0.2M	16	516	31	0	0	16	1
YouCook2 [56]	×	✓	✓	2018	@30fps15.8M	2,000	13,829	89	0	0	2 K	N/A
VLOG [14]	×	✓	✓	2017	37.2M	114 K	0	0	0	0	10.7 K	N/A
Charades [42]	×	×	✓	2016	7.4M	9,848	67,000	157	0	0	N/A	267
Breakfast [28]	×	✓	✓	2014	3.0M	433	3078	50	0	0	52	18
50 Salads [44]	×	×	×	2013	0.6M	50	2967	52	0	0	25	1
MPII Cooking 2 [39]	×	×	×	2012	2.9M	273	14,105	88	0	0	30	1

Table 3.1: Comparative overview of relevant datasets (action classes with > 50 samples)

videos have been collected of humans performing daily tasks at home. The problem with this dataset is that the actions recorded were scripted, meaning that the actor had a text in which he was asked to perform some steps. In this way the actions lose their naturalness, their inbred evolving and multi-tasking properties.

To solve these problems they decided to focus on first-person vision, such that the recording would not interfere with the actor's actions, increasing the possibility of a successful recording. Also, the viewpoint given by first-person vision allows us to record multi-task actions and the many different ways to perform a variety of important everyday tasks. In Table 3.1 we report a summary of the datasets compared by the authors.

3.1.2 Data collection

The data collection involved 32 people in 4 cities in different countries (in North America and Europe): 15 in Bristol/UK, 8 in Toronto/Canada, 8 in Catania/Italy and 1 in Seattle/USA between May and Nov 2017. Participants were asked to record each time they visited the kitchen for three consecutive days, starting filming just before entering the kitchen and stopping before leaving it. They participated in the process of their own free will without being paid in any way.

A few requests were asked of them. The first was to be in the kitchen alone during the recording, such that no inter-person interaction could interfere. The second one instead was to remove all items that could disclose their identity, for example, portraits or mirrors. In this way, they could remain anonymous.

Each participant was equipped with a head-mounted camera with adjustable mounting such that it could be adapted to the participant's height and possibly different environment. They had to check, before each recording, the battery life and the viewpoint, such that their stretched hand were approximately located in the middle of the camera frame. The camera settings were set for most of the videos to a linear field of view, using 59.94fps as frame rate and Full HD resolution

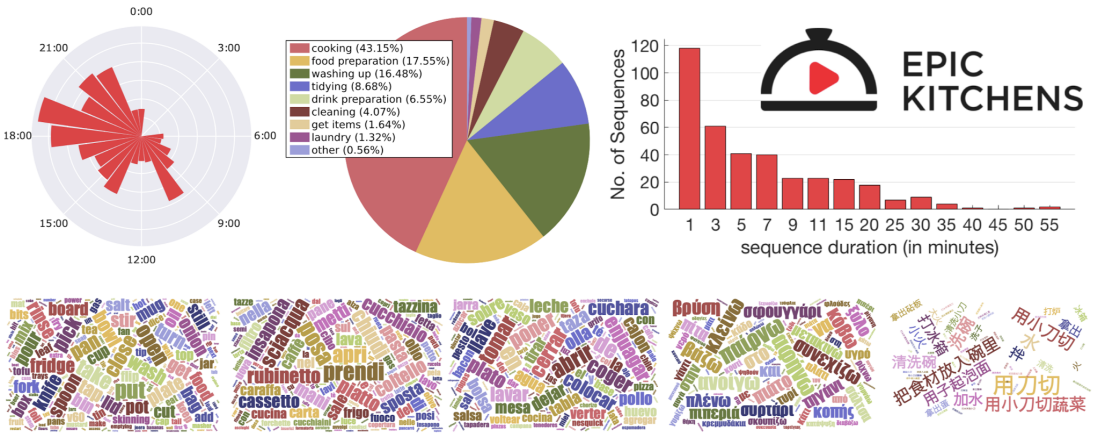

Figure 3.1: Top (left to right): time of day of the recording, a pie chart of high-level goals, histogram of sequence durations and dataset logo; **Bottom**: Wordles of narrations in native languages (English, Italian, Spanish, Greek and Chinese). Figure from [14].

of 1920x1080, however some subjects made minor changes like wide or ultra-wide FOV or resolution. In particular, 1% of the videos were recorded at 1280x720 and 0.5% at 1920x1440. Also 1% at 30fps, 1% at 48fps and 0.2% at 90fps.

On average, each participant recorded 13.6 sequences, each of those lasting on average 1.7 h while the maximum duration recorded was 4.6h. The duration of the recording was linked to the person's kitchen engagement. In Figure 3.1 we can see some statistics of the data acquired.

3.1.3 Data Annotation pipeline

After the end of a sequence, each participant was asked to watch the recording and narrate verbally the actions carried out to a microphone. The sound narration was chosen because it was faster than a written one, and participants were thus more willing to provide these annotations. The guidelines for narrations are reported in Figure 3.2.

The most used language was English, but other languages were used, if the participant was not so fluent in English. In particular a total of 5 languages were used: 17 people narrated in English, 7 in Italian, 6 in Spanish, 1 in Greek, and 1 in Chinese.

The motivation to obtain the narrations directly from the actors was because they surely knew what they were doing, avoiding misinterpreting some possible actions. The posthumous narration was instead motivated by the fact that actors

Use any word you prefer. Feel free to vary your words or stick to a few.
Use present tense verbs (e.g. cut/open/close).
Use verb-object pairs (e.g. wash carrot).
You may (if you prefer) skip articles and pronouns (e.g. “cut kiwi” rather than “I cut the kiwi”).
Use propositions when needed (e.g. “pour water into kettle”).
Use ‘and’ when actions are co-occurring (e.g. “hold mug and pour water”).
If an action is taking long, you can narrate again (e.g. “still stirring soup”).

Figure 3.2: Narration Guidelines are given to each participant to be followed after the completion of a recording. Figure taken from [14]

0:14:44.190,0:14:45.310	0:00:02.780,0:00:04.640	0:04:37.880,0:04:39.620	0:06:40.669,0:06:41.669	0:12:28.000,0:12:28.000	0:00:03.280,0:00:06.000
pour tofu onto pan	open the bin	Take onion	pick up spatula	pour pasta into container	open fridge
0:14:45.310,0:14:49.540	0:00:04.640,0:00:06.100	0:04:39.620,0:04:48.160	0:06:41.669,0:06:45.250	0:12:33.000,0:12:33.000	0:00:06.000,0:00:09.349
put down tofu container	pick up the bag	Cut onion	stir potatoes	take jar of pesto	take milk
0:14:49.540,0:15:02.690	0:00:06.100,0:00:09.530	0:04:48.160,0:04:49.160	0:06:45.250,0:06:46.250	0 :12:39.000,0:12:39.000	0:00:09.349,0:00:10.910
stir vegetables and tofu	tie the bag	Peel onion	put down spatula	take teaspoon	put milk
0:15:02.690,0:15:06.260	0:00:09.530,0:00:10.610	0:04:49.160,0:04:51.290	0:06:46.250,0:06:50.830	0:12:41.000,0:12:41.000	0:00:10.910,0:00:12.690
put down spatula	tie the bag again	Put peel in bin	turn down hob	pour pesto in container	open cupboard
0:15:06.260,0:15:07.820	0:00:10.610,0:00:14.309	0:04:51.290,0:05:06.350	0:06:50.830,0:06:55.819	0:12:55.000,0:12:55.000	0:00:12.690,0:00:15.089
take tofu container	pick up bag	Peel onion	pick up pan	place pesto bottle on table	take bowl
0:15:07.820,0:15:10.040	0:00:14.309,0:00:17.520	0:05:06.350,0:05:15.200	0:06:55.819,0:06:57.170	0:12:58.000,0:12:58.000	0:00:15.089,0:00:18.080
throw something into the bin	put bag down	Put peel in bin	tip out paneer	take wooden spoon	open drawer

Figure 3.3: Extracts from 6 transcription files in .sbv format. Figure from [14].

could perform their actions most naturally, without being concerned about labeling.

The second step of annotations consists of the transcription of the speech narrations. After testing some automatic audio-to-text algorithms, which led to inaccurate transcriptions, they opted for manual transcriptions and translation via Amazon Mechanical Turk (AMT), a crowdsourcing marketplace that allows them to do tasks that computers are still unable to complete. More in-detail requests to AMT are called HIT (Human Intelligence Tasks). To ensure consistency, the authors divided speeches into chunks of around 30 seconds by also removing silent parts and sent each chunk 3 times as HIT. In this way, they selected just HIT which had a correspondence. An example of transcription is shown in Figure 3.3

In the end, they collected 39,596 action narrations, corresponding to a narration every 4.9 seconds in the video. These narrations gave them a good starting point for labeling all actions with a rough temporal alignment, obtained from the timestamp of the audio narration Concerning the video, they still were not perfect. In fact:

- The narrations can be incomplete. So only narrated action will be considered in evaluation.
- The narration can be belated after the action takes place.
- The narration consists of participants’ vocabulary and free language. Similar terms have been grouped into minimally overlapping classes.

	ClassNo (Key)	Clustered Words
VERB	0 (take)	take, grab, pick, get, fetch, pick-up, ...
	3 (close)	close, close-off, shut
	12 (turn-on)	turn-on, start, begin, ignite, switch-on, activate, restart, light, ...
NOUN	1 (pan)	pan, frying pan, saucepan, wok, ...
	8 (cupboard)	cupboard, cabinet, locker, flap, cabinet door, cupboard door, closet, ...
	51 (cheese)	cheese slice, mozzarella, paneer, parmesan, ...
	78 (top)	top, counter, counter top, surface, kitchen counter, kitchen top, tiles, ...

Figure 3.4: Example of verbs and nouns classes clustered in wider groups. Figure taken from [14]

It is worth adding a few words about verb and noun annotations. Due to the freedom of terms and language, a variety of verbs and nouns have been collected. To reduce the number of them, they grouped these into classes with minimal semantic overlapping. More in detail, as regards verbs, they tried using automatic tools to cluster them but ended up manually clustering, due to the inefficient results; on the other hand nouns semi-automatically cluster them, preprocessing the compound nouns e.g. "pizza cutter" as a subset of the second noun e.g. "cutter" and also manually adjusting the clustering, merging the variety of names used for the same object, e.g. "cup" and "mug". In total, they obtained 125 verb classes and 331 noun classes. In Figure 3.4 we can see some examples of grouped verbs and nouns into classes, while in Figure 3.5 the authors show the verb classes ordered by frequency of occurrence in action segments, as well as the noun classes ordered by a number of annotated bounding boxes.

In addition to verb and noun annotations they also provide active object bounding box annotations. Similarly to verbs and nouns, they use AMT also for this task. Where each HIT aims to get an annotation for one object, for a maximum duration of 25 seconds, which corresponds to 50 consecutive frames at 2fps. The annotator can also state that the object is inexistent in at frame *f*. In total they collected 454,255 bounding boxes, some examples are provided in Figure 3.6.

3.1.4 Benchmarks and baseline results

The introduction of a new video dataset implies a variety of potential challenges that were not available before. Some of these are routine understanding, activity recognition, and object detection. To spur the beginning the authors define the previously stated three challenges, providing baseline results. In the following paragraphs we provide an overview of the challenges.

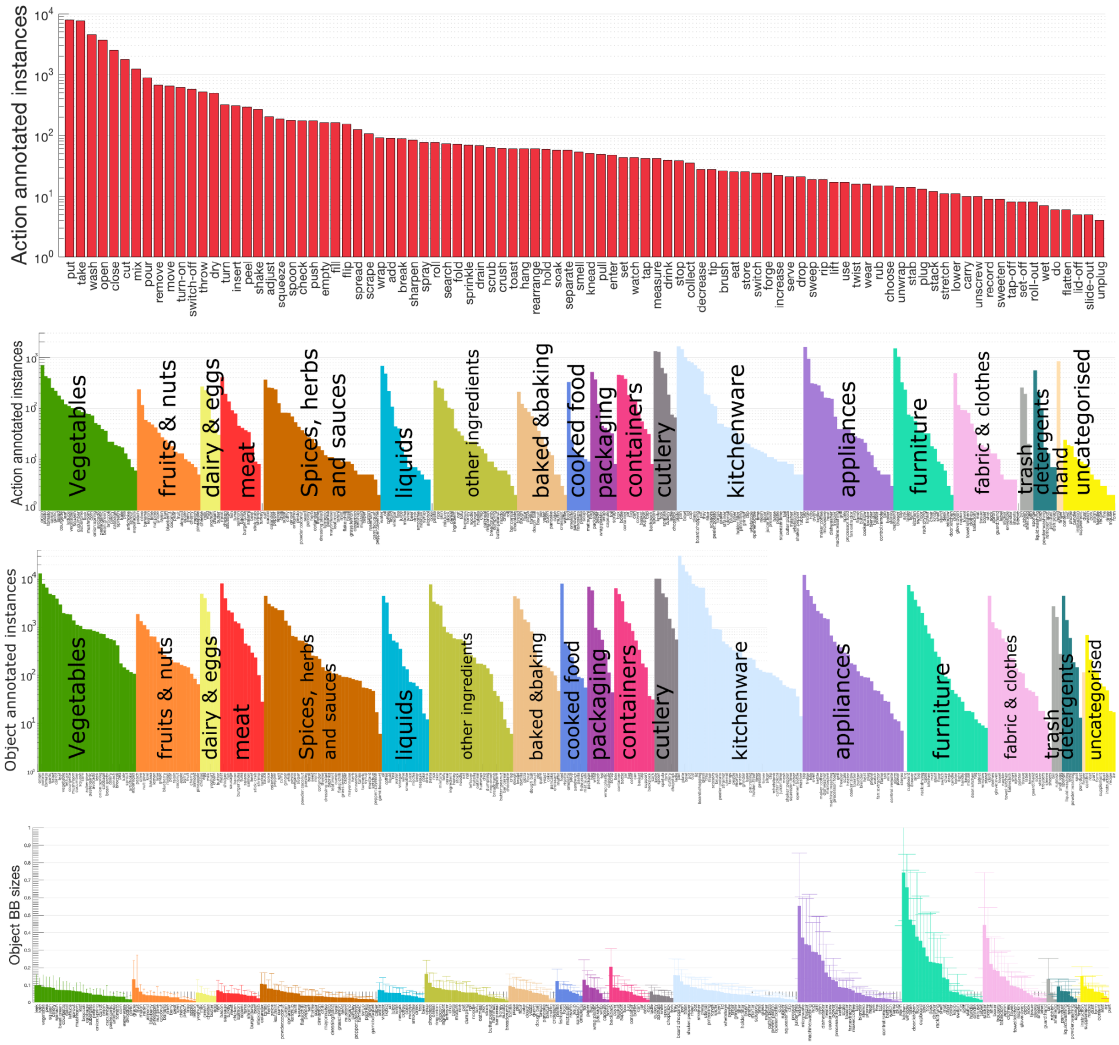

Figure 3.5: From Top: Frequency of verb classes in action segments; Frequency of noun clusters in action segments, by category; Frequency of noun clusters in bounding box annotations, by category; Mean and standard deviation of bounding box, by category. From [14]

Action recognition challenge Provided a trimmed action segment, the challenge requires to recognize what action class is performed, detecting the pair of verb and noun classes that compose the action. To participate in the challenge is asked to test the model on both splits[1] and for each test segment report the confidence

[1]To test the generalizability to novel environments they structured the test set to have a collection of *seen* and *unseen* kitchens.

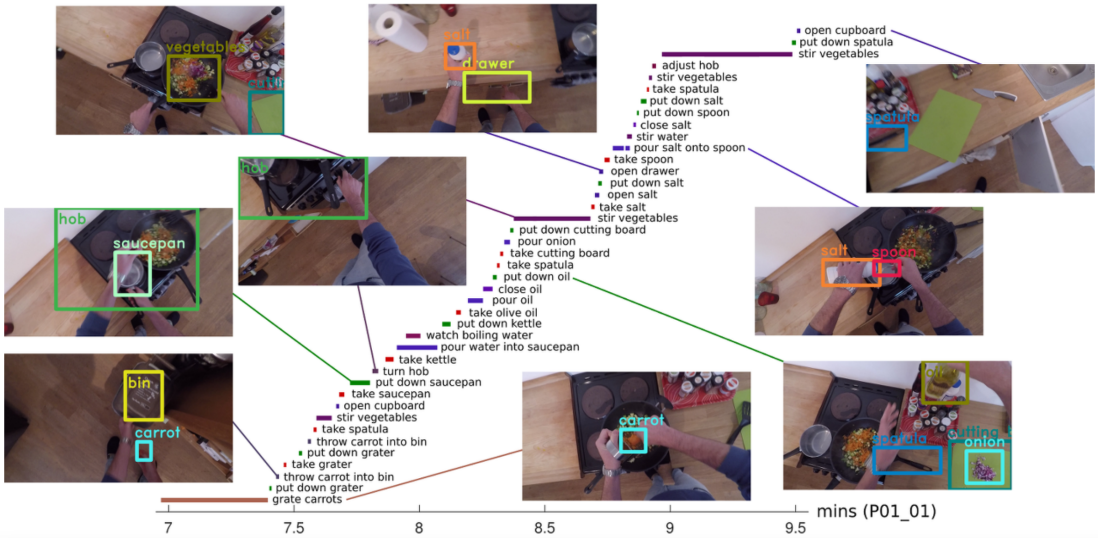

Figure 3.6: Sample of consecutive action segments with keyframe object annotations from [14]

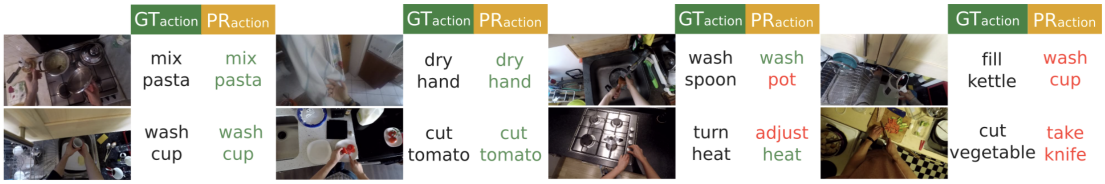

Figure 3.7: Sample qualitative results from the challenge's baseline of the Action Recognition Task

scores for each verb and noun class. In Figure 3.7 is reported a qualitative example of the task.

Action anticipation challenge Provided an anticipation time, which is 1s before the action starts, the challenge consists of classifying the future action into its action class composed of the pair of verb and noun classes. To participate in the challenge is asked to test the model on both splits and for each test segment report the confidence scores for each verb and noun class. In Figure 3.8 is reported

- **Seen Kitchens (S1):** In this split each kitchen is seen in both training and testing.
- **Unseen Kitchens (S2):** This divides the participants/kitchens so all sequences of the same kitchen are either in training or testing.

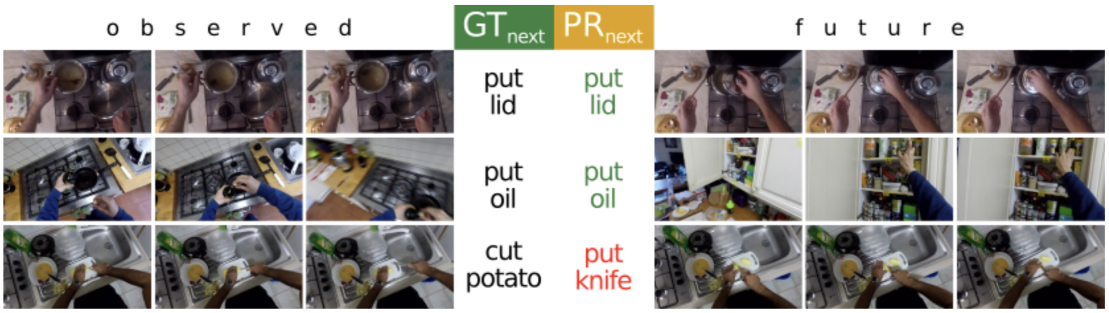

Figure 3.8: Sample qualitative results from the challenge's baseline of the Action Anticipation Task

Figure 3.9: Sample qualitative results from the challenge's baseline of the Object Detection Task

a qualitative example of the task.

Object Detection Challenge This challenge is required to perform object detection and localization. It must be noted that the annotations captured only *active* objects, namely objects involved in the action. To participate is required to provide predicted bounding boxes and their confidence scores on both dataset splits. In Figure 3.9 a qualitative example is reported.

Dataset release

- Dataset sequences, extracted frames, and optical flow are available at:

  `http://dx.doi.org/10.5523/bris.3h91syskeag572hl6tvuovwv4d`

- Annotations, challenge leader-board results and updates and news are available at `http://epic-kitchens.github.io`

3.2 EPIC-KITCHENS 100

In 2021 with [66] a new pipeline is introduced to extend the EPIC-KITCHENS dataset. EPIC-KITCHENS-100 collects 100 hours, 20M frames, 90k actions in 700 variable-length videos, capturing long-term unscripted actions in 45 different environments using head-mounted cameras. Due to its novel annotation pipeline, which will be described more in detail later, more complete annotations of fine-grained actions are available, allowing the creation of new challenges such as action detection[2] , cross-modal retrieval (e.g. Audio-Based Interaction Recognition) and domain adaptation[3].

3.2.1 Motivation

The introduction of EPIC-KITCHENS has transformed egocentric vision, showcasing the unique potential of first-person views for action recognition and in particular hand-object interactions. To continue on this previously marked path they decided to enlarge EPIC-KITCHENS, maintaining the *unscripted* and *unedited* object interactions nature. The unscripted characteristics make the dataset result in an unbalance of data, with novel compositions of actions in new environments, making it a challenging dataset for domain adaptation.

The most important novelty is the new annotation pipeline which allows to obtain denser and more complete actions' annotations in the recorded videos, enabling different tasks on the same dataset.

3.2.2 Data collection

The additional videos were obtained by half of the previous participants, 16 persons, half of the 32 previously involved, and 5 new additional subjects. In the end, the total number of participants reached 37 and the different kitchens were 45.

The new request for the subjects was to record 2-4 days of their kitchen routine.

[2]Action detection involves both recognizing the action and localizing the temporal intervals and spatial regions where the actions occur in a video.

[3]Training on a domain, e.g. a specific kitchen, and test on another domain, e.g. a kitchen of a different person.

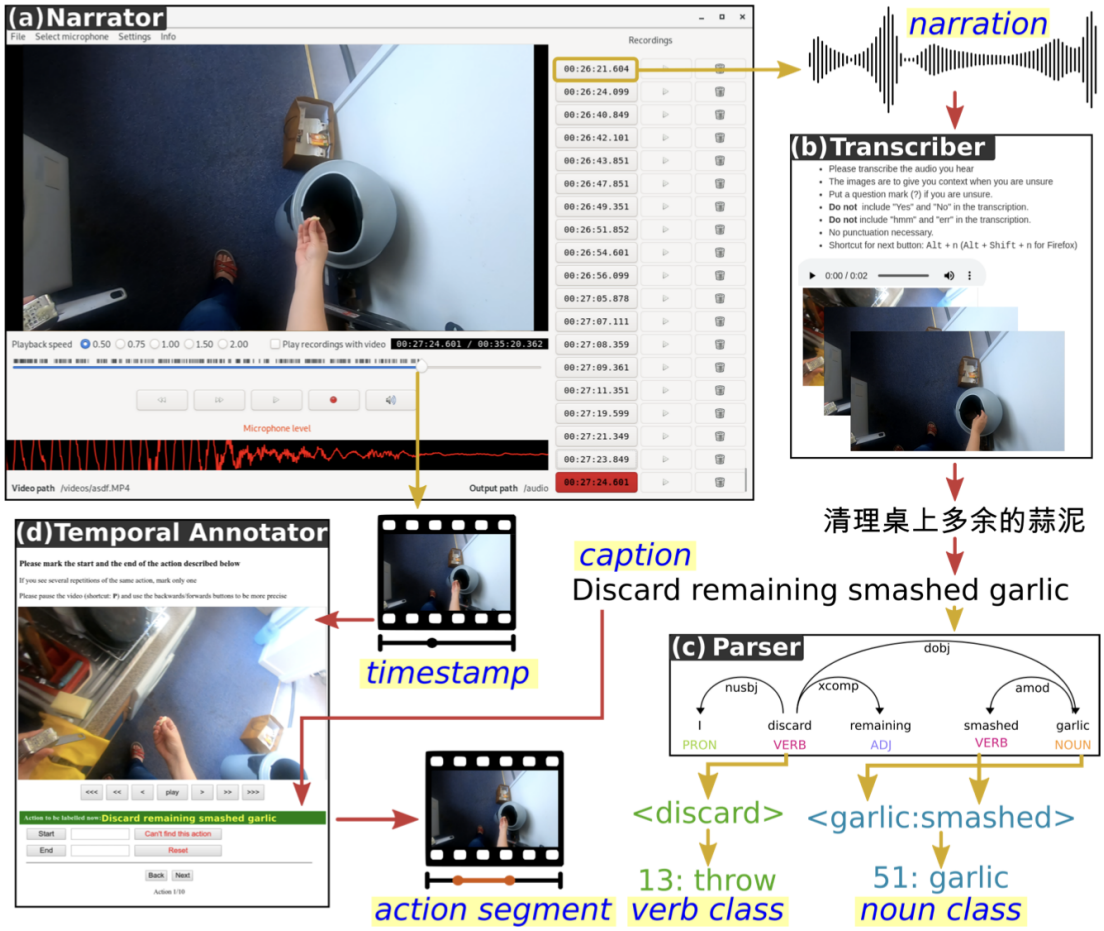

Figure 3.10: Annotation pipeline from [66]: **a** narrator, **b** transcriber **c** temporal segment annotator and **d** dependency parser. Red arrows show AMT crowdsourcing of annotations.

3.2.3 Annotation

An overview of the pipeline taken from the paper is reported in Figure 3.10.

Narrator The non-stop audio narration has been replaced with a *pause-and-talk* approach. By pausing the narrator can propose an initial temporal "pointing" but mostly avoids to miss or misspoke some actions due to lack of time. He does not have to narrate past actions while watching future actions, so short and overlapping actions are easier to annotate.

For this, an interface was built for the participants, which can be seen in Figure 3.10(a). An important new feature is the possibility to re-record and delete

a narration.

Transcriber Each narration is first transcribed and then translated into English by a hired translator for correctness and consistency. The transcription process has been facilitated by providing a new transcriber interface showing three images sampled around the time stamp. As a matter of fact, in the old EPIC-KITCHENS transcriptor struggled to understand some of the narration without any video context.

Each narration was analyzed by 3 AMT workers using a consensus of 2 or more workers. A transcription was rejected if its Word2Vec [67] embeddings were lower than a threshold of 0.9. In case of consensus failure, the transcription was selected manually.

Parser They used spaCy (`https://spacy.io`) to parse the transcriptions into verbs and nouns. Then they manually grouped those into minimally overlapping classes.

Temporal Annotator They built an AMT interface for the start/end times of action segments (see Figure 3.10.d). To improve the quality this time the number of workers was increased from 4 to 5.

Quality improvements The attention cared for during the annotation process led to denser and more accurate annotations. We can see the results by comparing the same action and their respective annotation from the two different pipelines in Figure 3.11.

3.3 EPIC-Fields

The necessity of suitable datasets and benchmarks in the unified problem of 3D geometry and video understanding, which has been pushed by Neural Rendering (See Section 2), led to the rise of EPIC Fields [68]. EPIC Fields is an expanded version of EPIC-KITCHENS comprehending 3D camera information. 96% of EPIC-KITCHENS videos were reconstructed, registering 19M frames in 99 hours recorded in 45 kitchens.

EPIC-KITCHENS is suited for studying the unified problem of geometric reconstruction and semantic understanding. As a matter of fact, egocentric videos are relevant to mixed and augmented reality applications which have spread in the last years, and the videos probe dynamic neural reconstruction due to their length (up to one hour) and their dynamic nature.

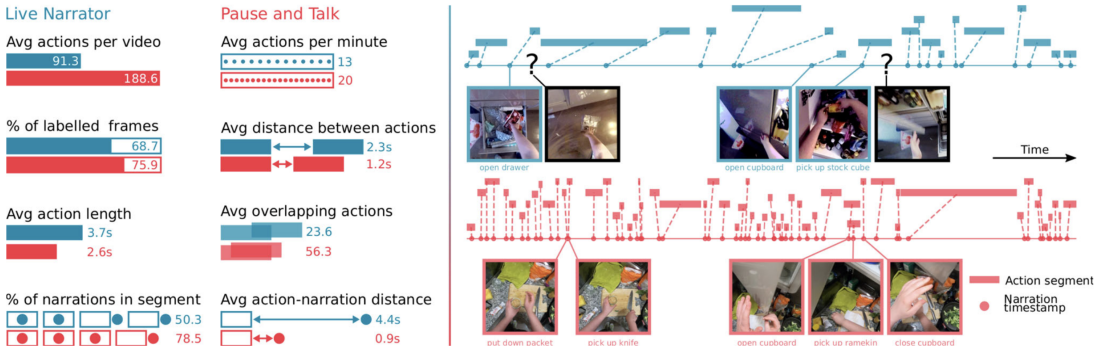

Figure 3.11: Comparing non-stop narrations (blue) to "pause-and-talk" narrations (red). Right: timestamps (dots) and segments (bars) for two sample sequences. "pause-and-talk" captures all actions including short ones. Black frames depict missed actions. Figure from [66].

Anyway obtaining camera information from EPIC-KITCHENS is difficult due to the complexity of its videos. Removing this step the authors try to ease the research by marrying 3D geometry to video understanding.

In conclusion, they made two contributions:

- Intelligent Subsampling of frames for SfM algorithms
- Introduce a set of benchmark tasks:
  - dynamic novel view synthesis: reconstruct the same scene from a different point of view.
  - identifying independently from the camera moving objects
  - segmenting independently from the camera moving objects
  - video object segmentation.

3.3.1 Data

Some of past egocentric datasets [69, 70] contain static 3D scans of the environment, separately reconstructed from the actions. This additional step is an additional expense both in time and money since the reconstructions are done with some dedicated costly hardware. In this work, they provide a pipeline to extract the geometric reconstruction of the scene by just processing the egocentric video. EPIC Fields extends EPIC-KITCHENS (See Section 3.2) to include camera pose information. For each frame camera, extrinsic and intrinsic parameters are provided, which enable tasks like 3D reconstruction. In total, they successfully processed 671 videos resulting in 18,790,333 registered video frames with estimated camera poses.

Motivation. The 3D reconstruction could help recognize different actions. Some actions could be located in the same 3D spot, e.g. washing the dishes at the sink. Also, the construction of this dataset could enable studying the relevance of 3D egocentric trajectories to actions (for anticipation), objects (for understanding object state changes) and hand-object understanding.

Collection Since EPIC-KITCHENS did not collect videos with 3D reconstruction in mind, its videos are difficult to reconstruct. Structure from Motion algorithms takes as assumption that the recorded scene is *static*, meaning that each object will always have the same position in 3D. However, the kitchen's activities involve the movement of objects like ingredients or utensils, and above all the presence of the operating hands.

Some other difficulties are introduced by:

- the **length of videos**: which on average last 9 mins
- the **skewed distribution of viewpoints**: the time spent in different parts of the scene is different. In particular, we have alternating phases of small motion around hot spots, e.g. washing dishes, and of fast motions, like taking something to finish some task.

The solutions to these problems were given by:

- **Intelligent subsampling** of video frames.
- Using **SfM** for reconstructing the filtered frames.
- **Registering remaining frames** to the reconstruction.

Subsampling This step aims to reduce the number of frames while keeping enough overlapping viewpoints for accurate reconstruction while diminishing the viewpoint skew. Overlap is measured,citing [68] *'by estimating homographies on matched SIFT features. Given a homography H, we define visual overlap r as the fraction of image area covered by the quadrilateral formed by warping the image corners by H. Windows is formed greedily, finding runs of frames (i+1,...,i+k) with overlap* $r \geq 0.9$ *to the first frame i. Filtering discards about 81.8 % of frames.*

Once filtered, frames are fed to COLMAP (Its functioning is reported in Section 1.2.5).

Dense reconstruction, automated verification, and restart The remaining frames are fed to COLMAP with initial reconstruction. The final reconstruction is accepted if over 70% of the frames are registered successfully. In the end, 631 videos were obtained.

In case of failure, the threshold r is increased,e.g. r≥0.95. This usually results in doubling the frames but increasing the success rate to 96%.

3.3.2 Benchmarks

The authors defined three new benchmarks to explore the combination of 3D and video understanding.

New-View Synthesis (NVS)

Given a reconstruction based on a subsample of frames, the goal is to predict new video frames based on their timestamps and camera parameters. The quality of the reconstruction is evaluated as proposed in [17], measuring the Peak-Signal-to-Noise Ratio (PSNR) of the reconstructed frames compared to the real ones, making the lack of a 3D ground truth irrelevant.

Video and Frame Selection. Due to the computationally expensive cost of Neural Reconstruction, they provided a benchmark of limited selection of videos, namely 50 for a duration of 14.7 hours and 2.86M registered frames.

Frame selection instead is needed to divide the data into train and evaluation splits. The evaluation frames were divided into three tiers of difficulty:

- **In-Action (Hard):** frames belonging to an annotated action segment. During an action, it is likely that objects in the scene are moved making it difficult to reconstruct.
- **Out-of-Action:** frames NOT belonging to an annotated action segment. These frames can be further divided into:
  - **Easy**: Frames for which exists a neighboring frame in the training set. The temporal proximity should ease the process of reconstruction.
  - **Medium**: Frames for which do not exist a neighboring frame in the training set.

Benchmark methods. Three different neural rendering techniques were used to illustrate their possibilities and limits in such challenging scenarios like EPIC Fields. The methods used were:

- **NeuralDiff [11]:** consists of three different NeRFs, each one tailored to a part of the scene: static background, moving foreground, and the actor. See Section 2 for more details.
- **NeRF-W [51]:** extend NeRF abilities by learning a low latent space that can modulate scene appearance and geometry. As a result, it can separate static and transient components.

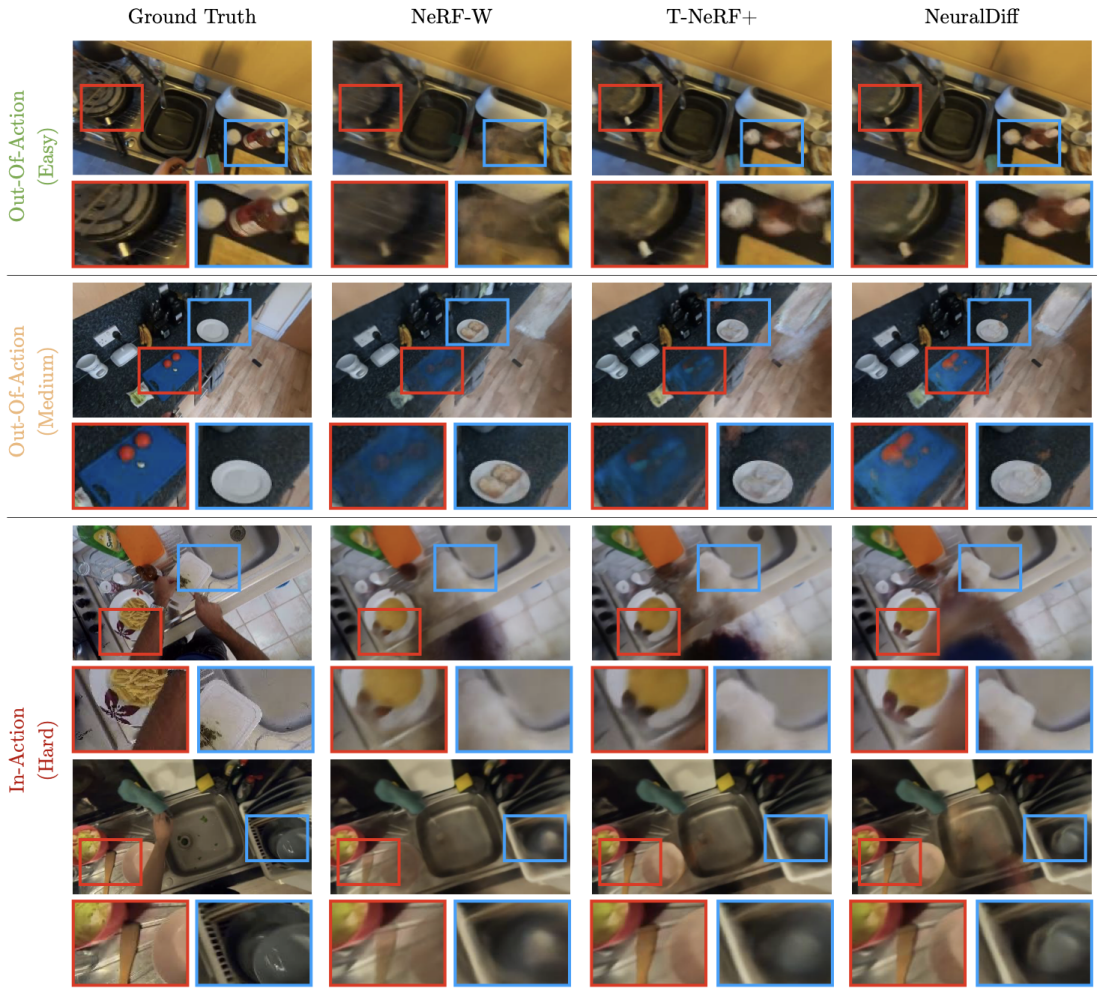

Figure 3.12: Dynamic New View Synthesis. Example from [68] of the output for the three different methods used: NeRF-W, T-NeRF+ and NeuralDiff. We can see how the initial labeling of difficulty for frames was accurate as the reconstructions struggle with Hard frames.

- **T-NeRF+ [71]:** time conditioned NeRF at which was added another NeRF to model the static background.

Unsupervised Dynamic Object Segmentation (UDOS)

In Unsupervised Dynamic Object Segmentation (UDOS) the objective is to find those regions in each frame that correspond to dynamic objects. The lack of a 3D ground truth makes 2D segmentation accuracy the only way to assess the model. In particular, mean average precision (mAP) was used, as proposed in [11].

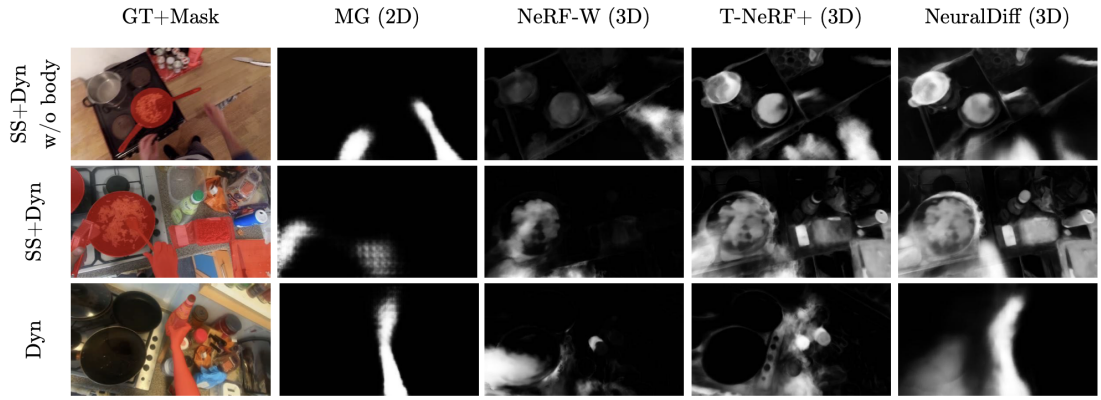

Figure 3.13: UDOS. Comparison,reported from [68], of the different methods' output. The 2D-based performs very well on dynamic objects, while 3D methods struggle a bit but can detect even semi-static objects.

Video and Frame selection. The used videos were the same as NVS, with the difference that only In-Action frames were considered, with VISOR annotations as ground truth. Actually, VISOR annotations were processed in the following way: the original masks were converted into foreground-background masks in three different ways, depending on the type of objects present.

- **Dynamic objects only** setting: a dynamic object is an object that is currently being moved by visible hands.
- **Dynamic and semi-static objects** setting: objects that moved, not necessarily in the current frame, are semi-static objects.
- **Dynamic and semi-static excluding body parts** setting: active hands are excluded, as some methods overfit to predicting hands solely as dynamic objects, ignoring other moving objects.

As Baseline methods, the authors used 4 methods: three based on 3D neural rendering techniques (NeRF-W, T-NeRF, NeuralDiff) and one based on 2D optical flow (Motion

Grouping (MG) [72]. The results are shown in Figure 3.13. It is worth noting how 3D methods are better at discovering semi-static objects. However none of the 3D methods explicitly consider motion and this can be seen as MG performs better on purely dynamic motion, due to the input being the optical flow.

Semi-Supervised Video Object Segmentation (VOS)

Semi-supervised video object segmentation is a standard video understanding task that consists of propagating some given masks for one or more objects in a reference

frame to the subsequent ones. Usually, this task is performed by 2D models but here the authors show how integrating the third dimension could be beneficial. The idea is to project the 2D mask in 3D, fixing its position in the 3D scene and reprojecting it depending on the new camera position.

Two baselines were provided, a 2D and a 3D one. In Figure 3.14 we can see the results and how the intuition previously described led to a good improvement.

Figure 3.14: VOS. Comparison of the two different methods taken from [68]. The 3D method is better at having as output something close to the ground truth. The 2D method instead is performing poorly.

3.4 VISOR

VISOR [73] is an extension of the EPIC-KITCHEN dataset introducing pixel annotations and a benchmark suite for segmenting hands and active objects. In particular segmentation annotations are provided in a *sparse* way, meaning that not every frame is segmented at pixel level.

The authors proposed a pipeline to obtain the annotations. Namely it consists of: (*i*) identifying *active* objects that are of relevance to the current action; (*ii*)annotating pixel-level segments via an AI-powered interface; (*iii*) relating objects spatially and temporally for short-term consistency. Some examples of annotations are reported in Figure 3.15.

They segment each *Active* object, namely each object that is involved in the current action. This is different, as we will see in later sections, from our goals, where we just want to segment *moving* objects. As shown in Figure 3.15, most of the time, when cooking the gas stove is segmented but it is not dynamic. Or as another example. if an object that is not involved in the current action is moved it would not be segmented (e.g. move a chair while getting something from the pantry).

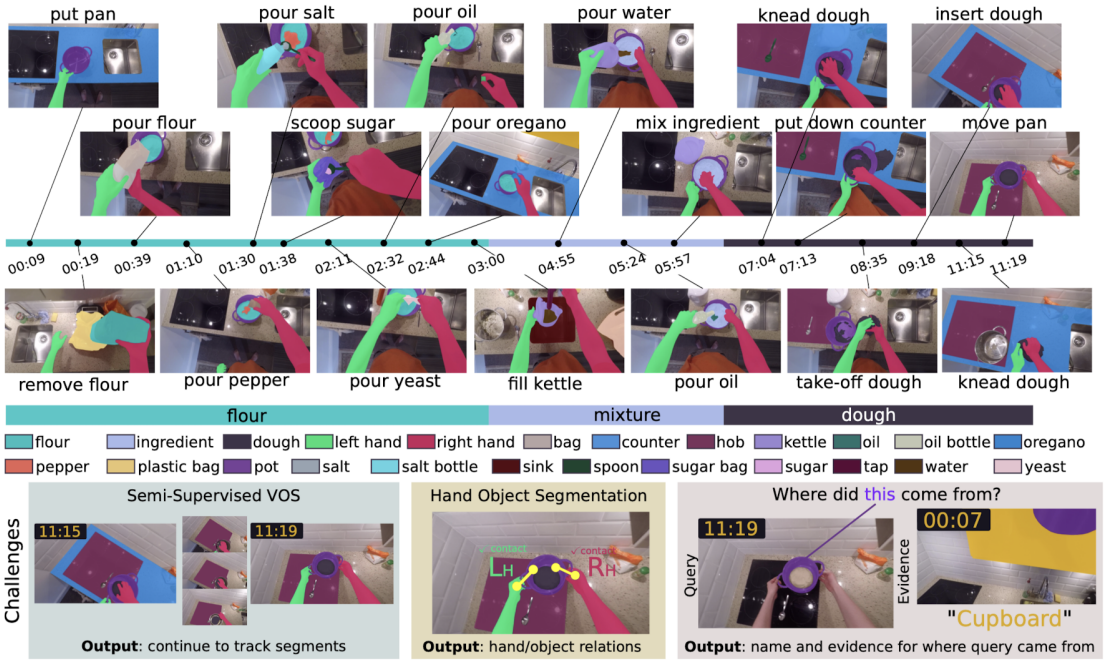

Figure 3.15: VISOR Annotations and Benchmarks. Sparse annotations of the P06-03 scene, where flour becomes dough in the end. Each color represents a different object. In the bottom part, the three proposed challenges are reported for the current scene. Namely: Semi-Supervised Video Object Segmentation (VOS), Hand Object Segmentation (HOS) and Where Did This Come From (WDTCF). Figure taken from [73].

3.5 Other egocentric datasets

3.5.1 Ego4D: Around the World in 3,000 Hours of Egocentric Video

Ego4D [74] is a massive-scale egocentric video dataset and benchmark suite. It captures 3670 hours of daily life activity located in a multitude of places (outdoor, workplace, leisure, etc.). The videos are captured by 931 different persons from 74 worldwide locations and 9 different countries. In addition to video frames, some scenes are accompanied by audio, 3D meshes of the environment, eye gaze, stereo, and/or synchronized videos from multiple egocentric cameras at the same event.

Furthermore, they presented new benchmark challenges centered around understanding the first-person visual experience in the past (querying an episodic memory), present (analyzing hand-object manipulation, audio-visual conversation, and social interactions), and future (forecasting activities).

Figure 3.16: Aria Glasses exploded in its components.

With this work, the authors aimed to push the research into first-person perception. All the data and other information can be found at `https://ego4d-data.org/`.

3.5.2 Aria Digital Twin: a new benchmark dataset for egocentric 3D machine perception

Aria Digital Twin (ADT) [75] is an egocentric dataset captured via Aria glasses (see Figure 3.16) with an additional object, environment, and human-level ground truth. It comprises 200 sequences of real-world actions performed by Aria wearers in two real indoor scenes. Each sequence is composed of: 1) raw data of two monochrome camera streams, one RGB camera stream, and two IMU (inertial measurement unit) streams; 2) sensor calibration; 3) ground truth data with 6-degree-of-freedom (6DoF) poses of the Aria glasses, object 6DoF poses, 3D eye gaze vectors, 3D human poses, 2D image segmentations, image depth maps, and 4) photorealistic synthetic renderings.

With their work, ADT researchers want to set a new baseline in the egocentric machine perception domain, which includes complex challenges like 3D object detection and tracking, scene reconstruction and understanding, sim-to-real learning, and human pose prediction; while also inspiring new machine perception tasks for augmented reality applications. Also, they want to promote the usage of ADT data, and to do this they created some benchmarks that demonstrate the usefulness of the data gathered.

Part II

Our Contribution

Chapter 4

Methodology

Our goal is to segment, and then remove, dynamic objects in 3D from egocentric videos and intelligently filter samples to remove redundant information. In this chapter, we present how we exploited the presented topics and used them to reach our goal (Section 4.1). We then describe in detail our proposed pipelines in Section 4.2.

4.1 Goals

3D dynamic object segmentation The main goal of this thesis is to Segment Dynamic Objects in 3D from egocentric videos and remove them from 3D reconstructions such that we have a clean reproduction of the recorded scene. Up to now, *Structure from Motion* (SfM) algorithms find it challenging to reconstruct dynamic scenes, resulting in messy point clouds, where the same object can appear multiple times within the scene. An explicative example is reported in Figure 4.1. A scene where a pizza is prepared from dough is reconstructed. It is clearly visible from the frames that the induction cooker is slowly proceeding the making of the pizza but in the reconstruction, it is reported in its integrity. This is due to a lack of temporal reasoning of structure from motion procedures. Also, above the pizzas there seem to be multiple pans intersecting one with each other. Obviously, this does not correspond to reality, but by moving of few centimeters at some time intervals, the process registers it as a new object each time it is in a new position. Other examples are visible like the chopping board and some objects on the inductor stove. In this example, our goal would be to reconstruct the kitchen cleaned of all the objects that moved during the video recording. This would be of immense potential since any environment could be reconstructed from just a video of it and the presence of dynamic objects/persons would not interfere with it.

Figure 4.1: Reconstruction of a pizza preparation video. The top and bottom frames give us a glance at the action performed during the video while the central pointcloud highlights the problems of SfM in dynamic environments like the superimposition of the same object on itself or the reconstruction of objects that are not always present in the scene, e.g. the two pizzas.

Optimization The second goal of this thesis is to reduce the computational times of the overall pipeline. As introduced in Section 2.3, our pipeline is based on *NeuralDiff* [11], a neural network that learns to represent a 3D scene by adjusting its weights. As this method is computationally expensive, our pipeline aims to speed it up through a filtering method that reduces the number of frames while keeping the same important information of *larger* samplings. This will be presented in Section 4.3. Also, we took a a simplified version of *NeuralDiff*, in which the actor, the person who is wearing the camera, the foreground, and the dynamic objects, are fused, since for our scopes, the distinction was not needed and thus we could remove one of the three neural radiance fields by accelerating the computations.

4.2 Pipelines

In this section, we present the actual methods that we considered and implemented to actually achieve our goals. As we have seen from Section I, 3D dynamic object segmentation is still an evolving field. We took inspiration from various works to

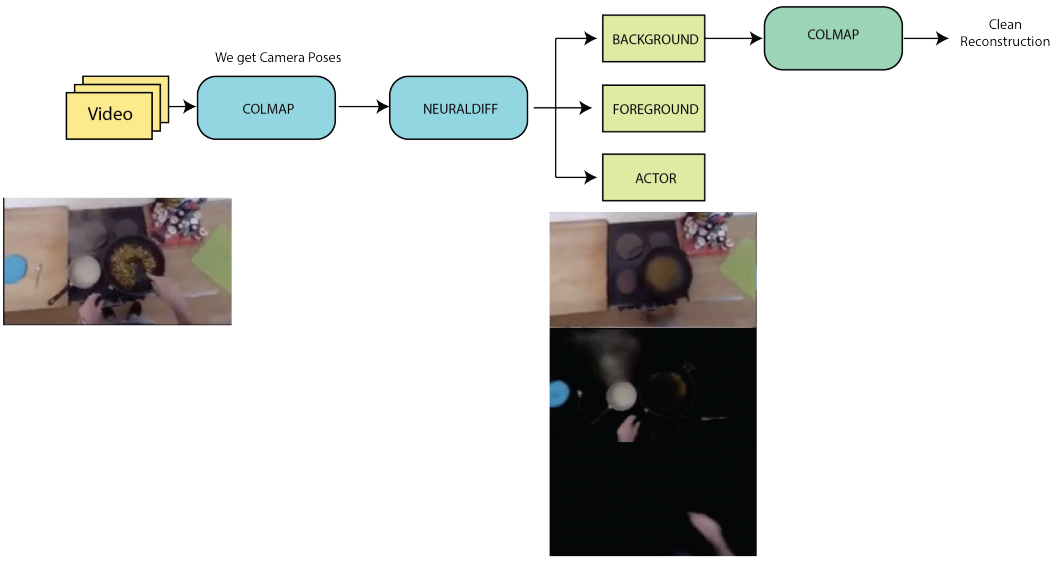

Figure 4.2: Basic Pipeline. In the basic pipeline a video (represented by the image of a person cooking) is reconstructed via COLMAP [15] and then fed to *NeuralDiff* [11]. At this stage, the frames are decomposed into actor, foreground, and background (as can be seen in the frames reported below *NeuralDiff*). A clean reconstruction is obtained by running another COLMAP step on the extracted background frames.

come up with some different ideas for actually segmenting 3D dynamic objects. All the methods revolve around a COLMAP reconstruction and a *NeuralDiff* [11] renderer. The basic block would in fact be *NeuralDiff* but a reconstruction of the scene with camera intrinsic and extrinsic parameters is required to correctly locate the images in the 3D space.

Colmap Pipeline The first and most trivial idea is to reconstruct the scene with the SfM algorithm, COLMAP, and then separate the static and dynamic objects in 2D on the frames using *NeuralDiff* [11]. Once the cleaning has been done we could reconstruct from the cleaned frames the static scene. In Figure 4.2 is reported the pipeline of this first approach.

Monocular Pipeline Yet, the previous method is not optimal because we have two COLMAP steps, which are computationally expensive, and one of *NeuralDiff*, which is computationally expensive too. In order to improve it we present a second pipeline that we call *Monocular* pipeline. This pipeline removes the last COLMAP

step by using a pre-trained neural monocular depth estimator [16]. This last module extracts pixels depth from each frames, then, using geometric projections, we obtain the 3D projection, if camera instrinsic and extrinsic parameters are provided.

Recalling Section 1.1, if we have a depth map Z and z_i the depth of the i pixel, we can project it in the 3D camera's coordinate system using the following relations:

$$\begin{cases} x_c = (u - c_x)\frac{z_i}{fx} \\ y_c = (v - c_y)\frac{z_i}{fy} \\ z_c = z_i \end{cases} \tag{4.1}$$

where u and v are respectively the pixel coordinates and c_x and c_y are the principal point coordinates. We can then move on to the world reference system by using the camera extrinsic parameters, that encapsulate the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ in itself:

$$Extrinsic = \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{t}_{3x1} \\ 0_{1x3} & \mathbf{1}_{1x1} \end{bmatrix}$$

obtaining the final coordinates as:

$$\begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} = \mathbf{R} \begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} + \mathbf{t} \tag{4.2}$$

In this way, we could train *NeuralDiff*, after obtaining a reconstruction of the scene included, and project the dynamic-segmented pixels into 3D space. Then to segment dynamic objects in 3D we take all the points of the COLMAP pointcloud that are at a distance less than a predefined value from the dynamic-projected points and classify them as moving. We remark that the projection of the pixels in the 3D space does not correspond to the COLMAP reconstruction as they are the result of two different processes. If we consider p_{pixel} a point projected from dynamic pixels to the world coordinate system and p_{col} a point in the world coordinate-system belonging to the initial reconstruction, we can label it following these relations:

$$p_{col} = \begin{cases} dynamic & \text{if } d(p_{pixel}, p_{col}) \leq \varsigma \\ static & \text{if } d(p_{pixel}, p_{col}) \geq \varsigma \end{cases} \tag{4.3}$$

where $d(\cdot, \cdot)$ is an euclidean distance and ς represent the predefined threshold. Reconstruction points arising from moving objects are expected to be in the same positions or at a close distance from the dynamic-projected ones. The same could be done with static points. This second pipeline is reported in Figure 4.3.

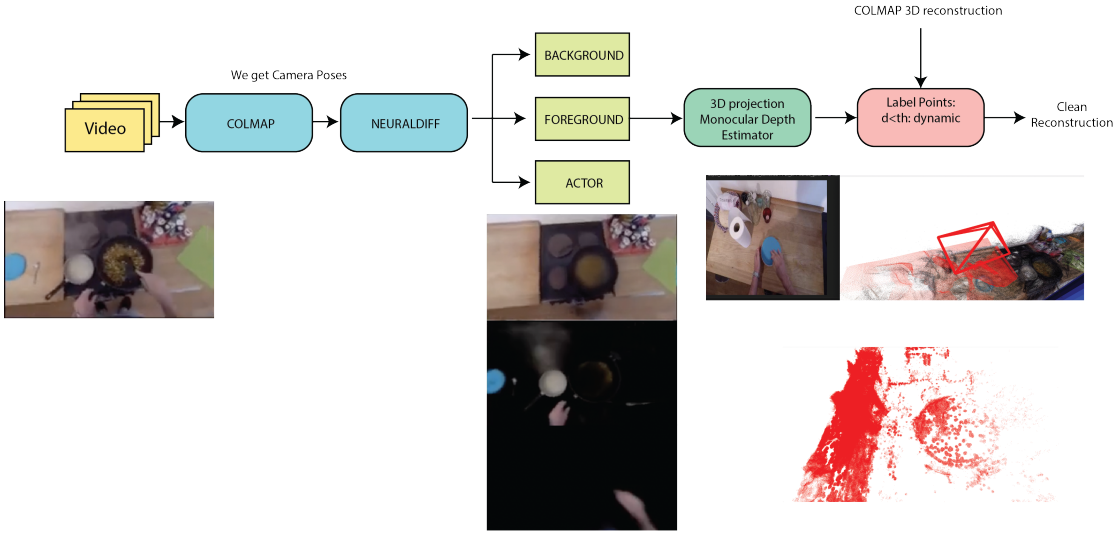

Figure 4.3: Monocular Pipeline. The monocular pipeline shares the first part with the basic pipeline. A video is reconstructed via COLMAP and fed to *NeuralDiff*. The difference is that here the dynamic layers are projected in 3D and points closer than a distance ς are segmented as dynamic. The red dots represent the pixel projected in the scene in the top image while in the bottom represents the dynamic labeled points of the COLMAP reconstruction.

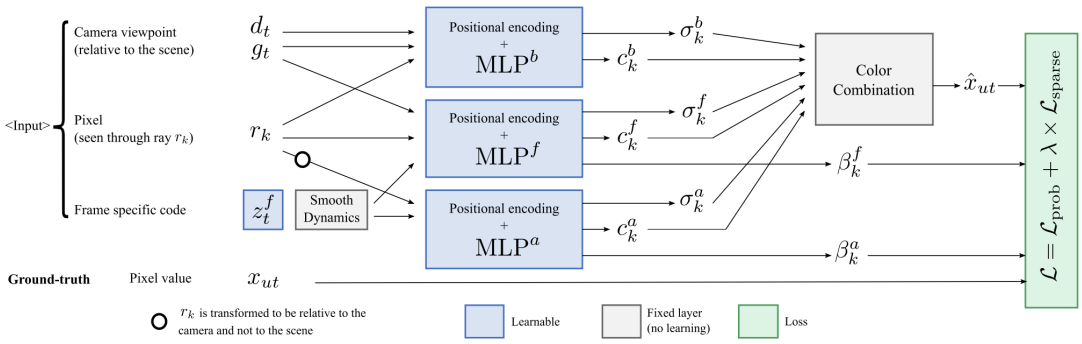

Figure 4.4: Overview of NeuralDiff. Given only the camera viewpoint g_t and a frame specific code z_t^f (learned latent variable), the *NeuralDiff* three stream architecture learns to predict the color value of pixel x_{ut} by combining information coming from the static model of the background, and from two dynamic components, one for the foreground objects, and one for the actor. Figure reported from [11].

NeuralDiff Pipeline The last pipeline instead takes advantage of the intrinsic knowledge of *NeuralDiff*, removing the necessity of the last step, being that SfM or a depth extractor. We recall the functioning of *NeuralDiff* in Figure 4.4. Given

a video sequence x *NeuralDiff* works with single pixels x_{ut} extracted from each frame x_t. For each training step it is required to provide the camera viewpoint g_t, the viewing direction d_t, and the frame-specific code, which represents the time evolution of the video. The model then, via three different *Multi-Layer-Perceptrons* (MLPs), learns to map the position of a queried 3D point to its corresponding color c_k and opacity σ_k for each layer, namely static, foreground, and actor, as if it is seen from that specific camera. This is obtained by querying multiple points sampled on the ray cast from the camera to the pixel and by minimizing the reconstruction error $\|x_{ut} - \hat{x}_{ut}\|$ between the true pixel x_{ut} value with and the predicted one $\hat{x}_{ut}$, that is obtained as:

$$\hat{x}_{ut} = f_u(g_t, z_t) = \sum_{k=0}^{M} v_k(\sum_{p \in \mathcal{S}} w^p(T_k) c_k^p) \quad where \quad v_k = \prod_{q=0}^{k-1} \prod_{p \in \mathcal{S}} T_q^p \tag{4.4}$$

where $f_u(g_t)$ is the function that maps the viewpoint to the single pixel u, T_i is the probability of a photon to be transmitted, z_t is the frame specific-code and $\mathcal{S} = \{b, f, a\}$ is the components involved (e.g. background and foreground: $\mathcal{S} = \{b, f\}$). The factor v_k requires a photon to be transmitted from the camera to point r_k through different materials. The weights $w^p(T_k)$ mix the colors based on point densities:

$$w^p(T_k) = \frac{\sigma_k^p}{\sum_{q=1}^{P} \sigma_k^q}(1 - \prod_{q=1}^{P} T_k^q) \tag{4.5}$$

This means that after training we are already grasping the 3D structure of the scene. In addition, the scene is divided in its motion components by the MLPs and any additional step would be unnecessary. To segment and remove dynamic objects, we took the MLP that encodes the static part of the scene and queried the points that were reconstructed in the beginning. The static MLP should give opacity, or density, only to points that were still in the scene while the opacity of any point that belonged to any moving object should be close to zero, as they are encoded by the other MLPs. Any point p is then labeled as:

$$p = \begin{cases} dynamic & \text{if } \sigma^b \leq \tau \\ static & \text{if } \sigma^b \geq \tau \end{cases} \tag{4.6}$$

where σ^b is obtained from $(\sigma^b, c^b) = MLP^b(p, d_t)$, b in apex stays for *background*, and τ is a threshold to decide under which the opacity is relevant. In our tests we chose τ experimentally, looking how the varying threshold affected qualitatively te scene reconstruction.

The overall pipeline can thus be simplified as in Figure 4.5, and it is referred to as *NeuralDiff-Pipeline*.

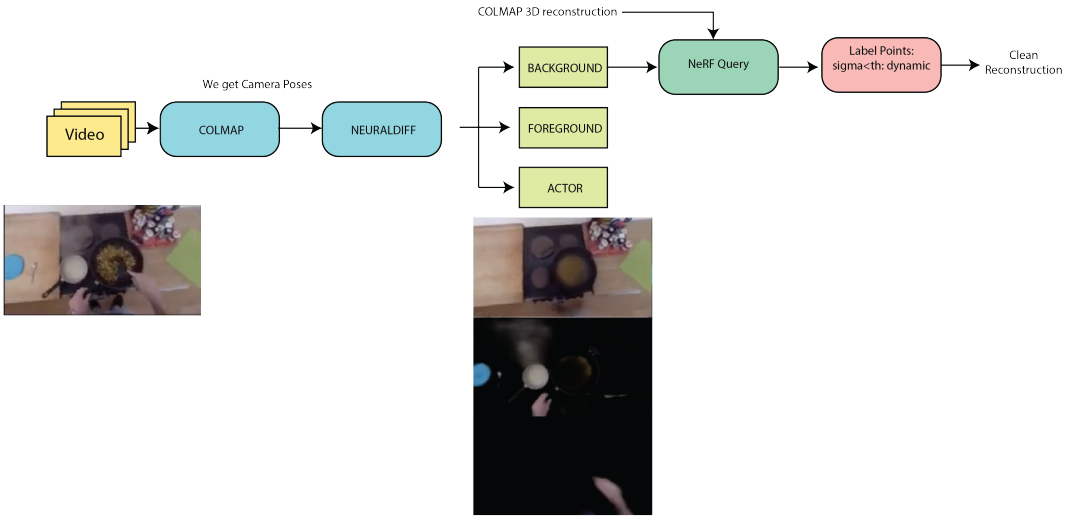

Figure 4.5: NeuralDiff Pipeline. In this pipeline we obtain the three motion layers as in the other methods but the segmentation is performed by querying the static neural renderer with the positions of the points belonging to the COLMAP reconstruction. Each point is segmented as a dynamic if its density is less than a predefined value.

NeuralCleaner The last modification that we made was to remove the distinction between the actor and the foreground, combining them into a single stream. The foreground MLP is thus now responsible for capturing every moving object. Since it was out of our scope to distinguish between these two layers,we reduced the computational times by removing one of the three neural streams of *NeuralDiff*. In Figure 4.6 we show the resulting architecture for the proposed model. This pipeline has been named *Neural Cleaner*. The decision process is the same as the *NeuralDiff* pipeline and can be seen in Equation 4.2.

4.3 Filtering

The filtering method consists of seeking temporal windows where frames are overlapped and keeping just a frame per window. The overlap is computed by estimating homographies [1] on matched SIFT (see Section 1.2) features. In Figure 4.8 some examples of overlaps are visualized.

[1]A homography is a transformation that maps points from one image to corresponding points in another image.

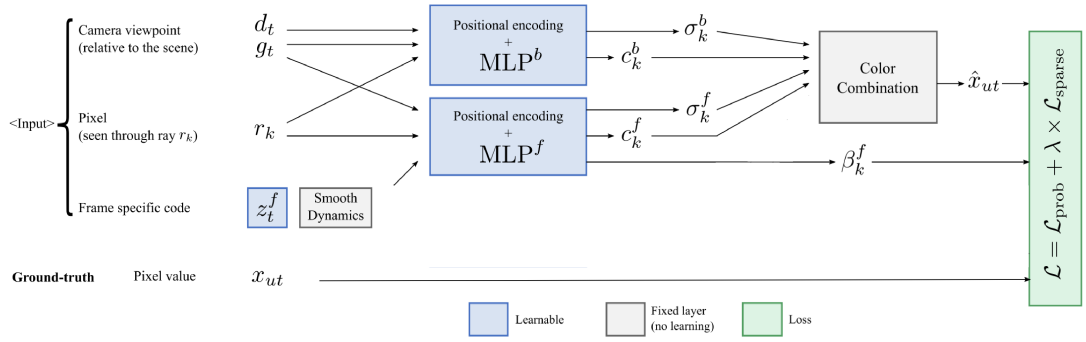

Figure 4.6: NeuralCleaner Architecture. Given only the camera viewpoint g_t and a frame specific code z_t^f (learned latent variable), the NeuralCleaner two-stream architecture learns to predict the color value of pixel x_{ut} by combining information coming from the static model of the background, and from the dynamic component, which is unique in this case. The parameters of this model are learned using a probabilistic loss $\mathcal{L}$.

EF Sampling. This technique was proposed in EPIC-Fields [68] to obtain accurate 3D reconstructions from egocentric videos, which present the challenging problems of dynamic objects, long duration video ($\sim$ 9min on avg) and the skewed distribution of viewpoints, namely the fact that in videos there are phases of slow motion around hot spots (e.g. around the gas stove) alternating to high motion in transition actions (e.g. taking something from the pantry). The main idea was to remove redundant frames while maintaining enough overlap and temporal coverage to allow an accurate reconstruction. We refer to this method as *EF-Sampling*.

Intelligent Sampling. We took inspiration from this technique for our sampling method that we called *Intelligent Sampling*. This sampling is used to keep only important frames, trying to remove any redundant information for the neural rendering step. The idea of maintaining important frames is different from the SfM step. In our case, an improved set of images would minimize overlap, ensuring comprehensive coverage of the scene. Ideally, if the images are limited in number, they should be evenly spaced throughout the scene for optimal representation. In this way, contrary to the idea of EF-Sampling reported in Paragraph 4.3, there should not be any blind spot and the environment is captured in its entirety. Its position in our pipeline is reported in Figure 4.7. The COLMAP reconstructed frames are intelligently sampled and fed to NeuralDiff.

The actual implementation is strictly based on EF-Sampling and since its goal is the contrary of the latter, they share part of the method. Intelligent sampling given a set of N frames performs an EF-sampling with an overlap threshold such that

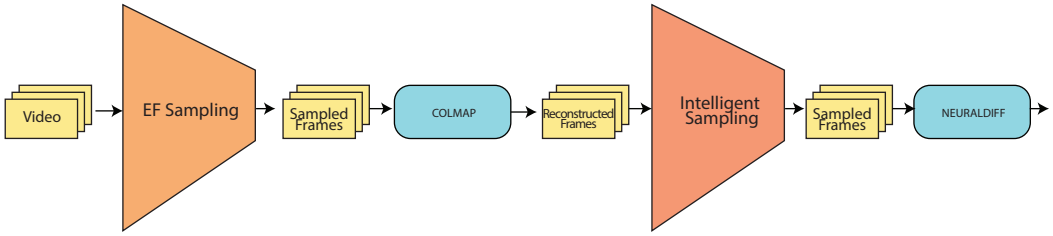

Figure 4.7: Sampling Steps in our pipelines. The initial video is subsampled by EF-Sampling and the resulting frames are fed to COLMAP. Once COLMAP reconstructed the scene these frames are again subsampled via Intelligent Sampling and fed to *NeuralDiff*.

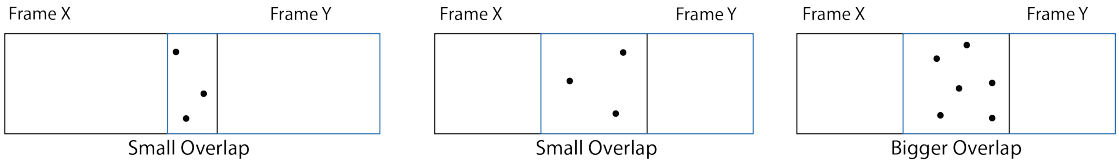

Figure 4.8: Example of overlapping frames (X and Y). The dot represent the features extracted. The left and central examples present the same level of overlap even though the image frames are closer together in the central example because the number of features shared is the same. The right examples instead present a higher level of overlapping due to the bigger number of features.

it gives $N - N_{desired}$ frames. These frames are then discarded and the remaining $N_{desired}$ frames are kept.

AU Sampling. We also tried a less rigid approach with the *AU Sampling*. This other method is a hybrid of Intelligent Sampling and uniform sampling (which is the baseline we tried to improve). In AU Sampling, we perform an Intelligent Sampling taking more than $N_{desired}$frames. The $N_{desired}$ frames are then uniformly extracted from this larger set. In this way, we keep some overlapping frames by increasing the threshold and then obtain equitemporal-spaced samples from the entire duration of the video.

Chapter 5

Experiments

In this chapter we will present: in Section 5.1 the data used for the actual testing of our proposed pipelines and the metrics we will be using to assess our results; in Section 5.3 the results from COLMAP reconstructions and some exploration on changing some experimental parameters, accompanied by some qualitative visualization; in Section 5.4 the Monocular pipeline results; in Section 5.5 our sampling strategy results compared with the Uniform one; and in Section 5.6 the *NeuralDiff* pipeline results.

5.1 Data selection

The data selection was dictated by our problem. Indeed evaluating 3D scenes reconstructions is not an easy task due to the lack of 3D ground-truths. These are usually very expensive due to costly hardware scanners but sometimes are also not available, as in our scenario, where we would like a static-dynamic segmentation. In our case obtaining the static part would mean to clean the scene from all the possible moving objects which adds an extra cost in terms of time, but in other scenarios 'cleaning' the environment could not be allowed.

EPIC-Diff. For this fact we evaluated our scene reconstructions on a subsample of the EPIC-KITCHENS extension proposed in *NeuralDiff* [11], known as EPIC-Diff. In this extension, the authors of *NeuralDiff* added manually pixel-wise segmented masks for ten scenes of which we just considered the P01-01, P03-04, P04-01, P09-02, P16-01, P21-01 scenes.

Figure 5.1: Two frames from EPIC-Diff and their corresponding manual dynamic (in white)/background (in black) segmentation masks.

5.2 Metrics

As regards metrics we looked in literature for a way to evaluate our results but unfortunately each method involved a ground truth which for our dataset is not available. Possible ways to obtain a groundtruth could be manual annotations or simulating the environments. Both these two methods would take a considerable large amount of time and are also beyond the scope of this thesis.

For this reason we ended up by using the metrics proposed in [11], exploiting their manually segmented frames as annotations, of which two examples are reported in Figure 5.1, where the white pixels corresponds to moving objects.

These are: *PSNR* and *mAP*.

5.2.1 PSNR: Peak signal-to-noise ratio

The Peak Signal-to-Noise Ratio (PSNR) [76] is a metric commonly used in image and video processing to quantify the quality of a reconstructed or processed signal, like an image or video. It gives a measures of the ratio between the maximum possible power of a signal (MAX) and the power of the distortion or noise that affects the signal (MSE).

The formula for PSNR is usually expressed in decibels (dB) and is given by:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{\text{MAX}}{\text{MSE}} \right)$$

where:

- MAX is the maximum possible pixel value of the image (1 in our case).
- MSE is the Mean Squared Error, which represents the average squared difference between the original signal and the reconstructed or distorted signal.

It is worth noting that a high PSNR does not guarantee that the processed signal will be perceived as visually pleasing or high-quality by humans, especially in the case of perceptually sensitive applications like image and video compression.

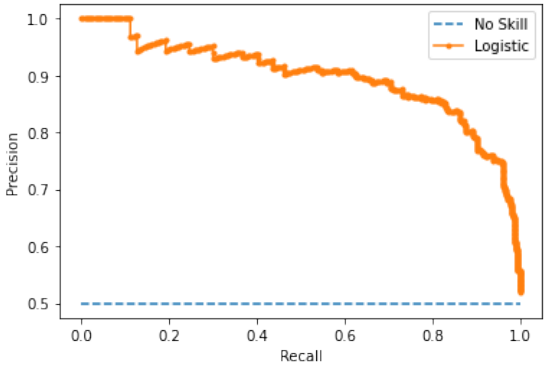

Figure 5.2: Example of Precision-recall curve. We can see how the bottom line model represents the worst a model can perform, e.g. predict every sample as it is coming from the same class,if the dataset is balanced. A better model would *tend* to the upper-right corner, which instead represents the best possible model, a model that have maximum precision and recall.

5.2.2 AP: Average Precision

Average Precision (AP) is a metric commonly used in object detection and information retrieval to evaluate the performance of machine learning models [77]. It measures the *precision-recall* trade-off of a model.

It can be useful to remind what *Precision* and *Recall* are. Namely:

$$Precision = \frac{TP}{TP + FP} \tag{5.1}$$

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

where:

- TP=True positive
- FP=False positive
- TN=True negative

Average precision is then computed as the area below the precision-recall curve, specifically the curve obtained by varying the confidence threshold of the inference model as shown in Figure 5.2. That is why it can also be found in literature as AUC (Area Under Curve). Its scalar value summarize the precision-recall performance of the model. A higher AP is desirable, indicating a model that effectively retrieves relevant instances while minimizing false positives.

5.3 COLMAP Reconstruction

Once we had fixed the data we were working on, we proceeded to do some experiments on COLMAP reconstructions. More precisely, we took scene P01-01 and tried varying both the number of frames and their resolution for the reconstruction. Most of the scenes have too many frames to handle, which could result in out-of-memory issues or at the least in a long computational time. We aimed to find a good compromise between *quality of reconstruction* and *computational time*.

5.3.1 Results

The first thing we did was to subsample the frames using the same technique reported in EPIC-Fields [68] and explained in Section 4.3. We report the results of the COLMAP reconstructions both quantitatively and qualitatively.

It is worth noting a few things watching these references.

Resolution The first one is that the resolution plays an important role in the successfulness of the reconstruction as we can see from Table 5.1, where the higher number of points is obtained at the highest resolution. The same subsample of frames is reported and the right one at a resolution of 114x64 failed. This is due to the feature extractor, which in a high-resolution image can retrieve information that instead is lost in low-resolution frames. A lack of significant features means no matching between images so the reconstruction has very few frames matched. The qualitative results of changing the resolution during the COLMAP reconstruction are shown in Figure 5.4, where we can see two different resolutions in comparison. It can be seen that a higher resolution allows a denser pointcloud.

Amount of frames The second thing is that the higher the frames the better. In Table 5.2 we can see on each row the distinct steps COLMAP performs during a reconstruction and their duration for each different subsample reported in each column (the columns' names are in the form "P01_01_xx" where xx represent the subsample, e.g. P01_01_08 is the subsample of scene P01_01 having 2598 initial frames). We can see that the number of points reconstructed is correlated to the number of the initial frames but also to the computational cost. Chances of matching increases and also we will have more areas of the environment covered, as shown in Figure 5.3. We can see that augmenting the number of frames more parts of the kitchen are revealed, *e.g.* the round table at the center of the room, the sideboard in front of the sink. But also some important objects that are visible in the video, like dishes on top of the table. This is a key point to the development of our pipeline because we need to be sure that the scene contains points deriving from the motion of objects.

Scene	P01_01_04	P01_01_04	P01_01_04
Initial Frames	1231	1231	1231
Reconstructed Frames	765	648	6
Resolution	456x256	228x114	114x64
PCD points	**629270**	152763	-
Duration	1h 7min 5s	44min 14s	-
Feature Extraction	16 s	7s	-
Exhaustive Matcher	42s	33s	-
Mapper	18min 35s	23min 57s	-
Image Undistorter	1s	0s	-
Patch Match Stereo	47min 1s	19min 28s	-
Stereo Fusion	30s	9s	-

Table 5.1: Comparison of recostruction of scene P01_01 using different resolutions. The higher the resolution, the better. Too low resolution, as 114x64 in this case can lead to a unsuccessful reconstruction.

Scenes	P01_01_04	P01_01_06	P01_01_08	P01_01_09
Initial Frames	1231	1487	2598	5223
Reconstructed Frames	648	911	2045	4741
PCD points	152763	204024	460914	1079375
Duration	44min 14s	1h 12min 34s	3h 6min 32s	10h 41min 8s
Feature Extraction	7s	8s	13s	28s
Exhaustive Matcher	33s	49s	2min 32s	10min 22s
Mapper	23min 57s	44min 5s	2h 1min 25s	8h 18s
Image Undistorter	0s	1s	1s	2s
Patch Match Stereo	19min 28s	27min 15s	1h 1min 18s	2h 24min 16s
Stereo Fusion	9s	16s	1min 3s	5min 42s

Table 5.2: Comparison of recostruction details for scene P01_01 using different initial frames at same resolution of 228x128. The higher the frames, the better the reconstruction but at a higher computational cost. The columns' names are in the form "P01_01_xx" where xx represent the subsample, e.g. P01_01_08 is the subsample of scene P01_01 having 2598 initial frames.

By considering these results and always keeping in mind the time of computation at our disposal we opted to feed the next pipeline with around five thousand frames at a resolution of 228x128. The pipeline of *NeuralDiff* is heavy and working at full resolution was prohibitive in the number of experiments we could try.

In Figure 5.5, we present visualizations of other kitchen reconstructions, each displaying the challenges of reconstructing egocentric videos due to multiple intersecting objects on the countertop.

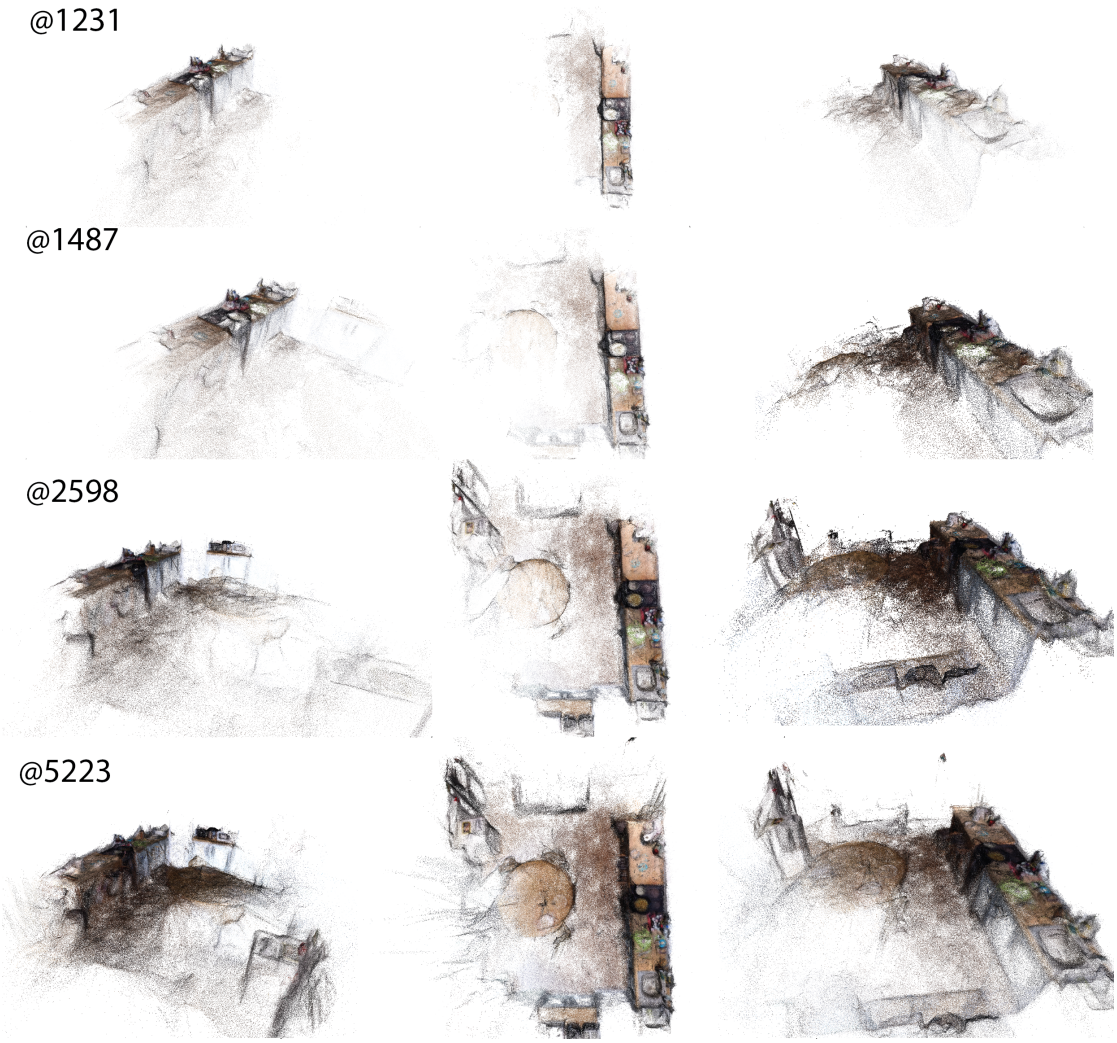

Figure 5.3: Different COLMAP pcd reconstructions changing number of samples, that is reported after the @. Each row is the same reconstruction viewed from different viewpoints. From top to bottom, the number of frames increases. We can see how the number of frames positively affects the reconstruction.

Figure 5.4: Different COLMAP pcd reconstructions changing resolution. Each row is the same reconstructions viewed from different viewpoints. The first reports the reconstruction for a resolution of 456x256 and the bottom one the half resolution. It is clear how the resolution has a beneficial impact on the overall reconstruction.

5.4 Monocular Pipeline

Here I report the qualitative result obtained from the Monocular Pipeline. As expected the results are really poor. The main reason for the failure of this technique is due to the inaccuracy of the depth estimator. In Figure 5.6 we can see on the right a frame (the name on top is in the form"Scene:@x" where scene represents the scene and x represents the number of the frame) and on its left, its pixel projection in the 3D reconstruction is reported in red, with the camera represented by the red frustum. We can notice that in some scenes it seems to capture the main elements, like the hands in scene P01-01 or the pot in scene P16-01. However, others like P09-02 seem to completely get it wrong, as the smartphone is not visible in the projection.

Once the frames are projected in the environment space we also have to find a threshold for the distance at which a reconstruction point is labeled as dynamic or static. This makes the pipeline highly scene-specific requiring each time a lot of fine-tuning for a mediocre result.

Also using a distance principle for segmentation, we can see how the scene deteriorates in this form of spherical groups of points (see Figure 5.7). In Figure 5.8 we can see how the segmentation of what is dynamic and what is static change depending on the distance threshold. In particular, the scene taken was P03-04 and the values reported are 0.5, 5, and 20.

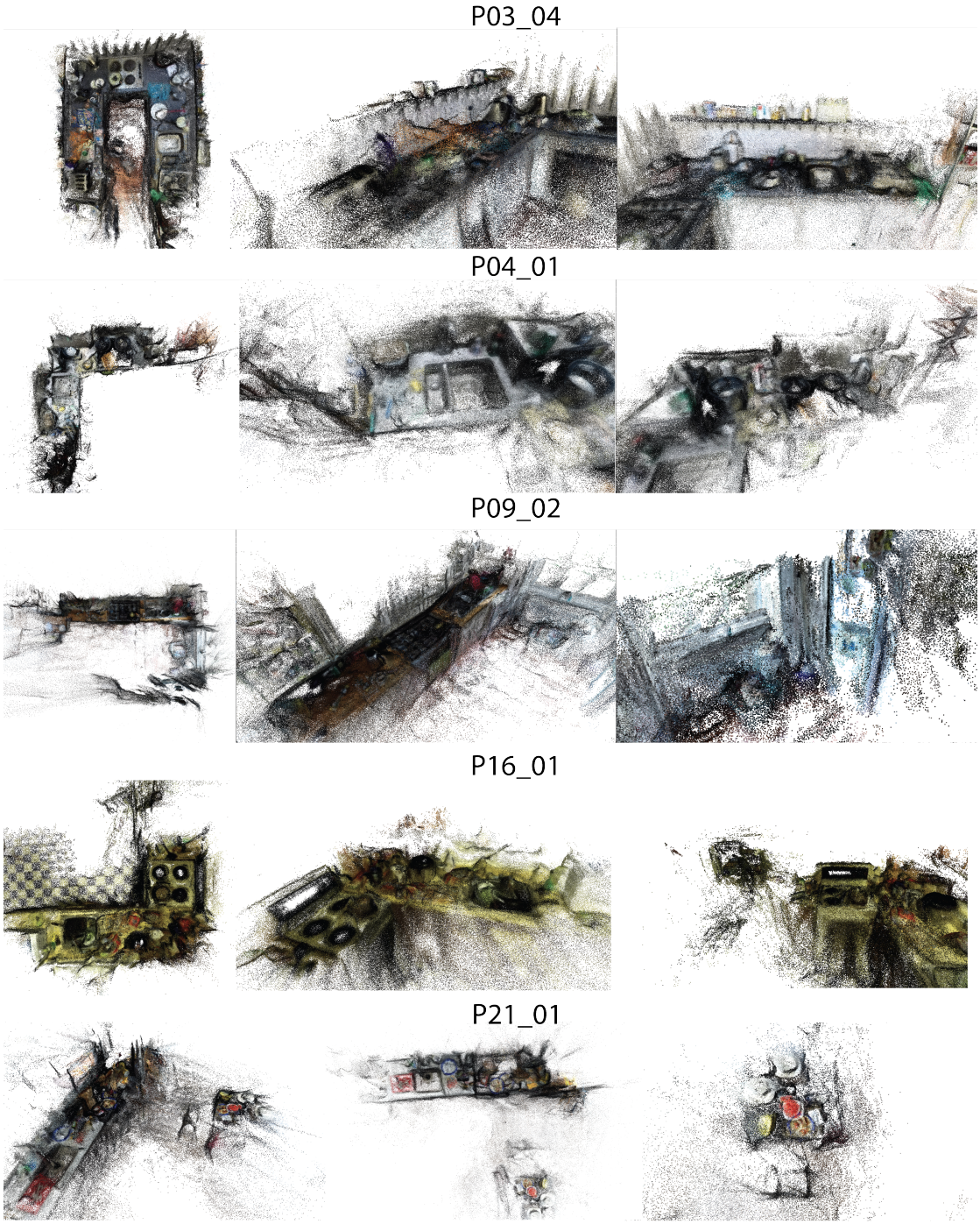

Figure 5.5: COLMAP reconstructions for scenes P03-04, P04-01, P09-02, P16-01,P21-01. Each row represents a different kitchen.

Figure 5.6: Different scenes where monocular depth estimation was performed. In particular on the right we can find the frame that is instead projected (in red) on the left in the 3D reconstruction of that kitchen. The red frustum (pyramid) represents the camera position and orientation in the space. The name on top is in the form "Scene:@x" where scene represents the kitchen and x represents the number of the frame.

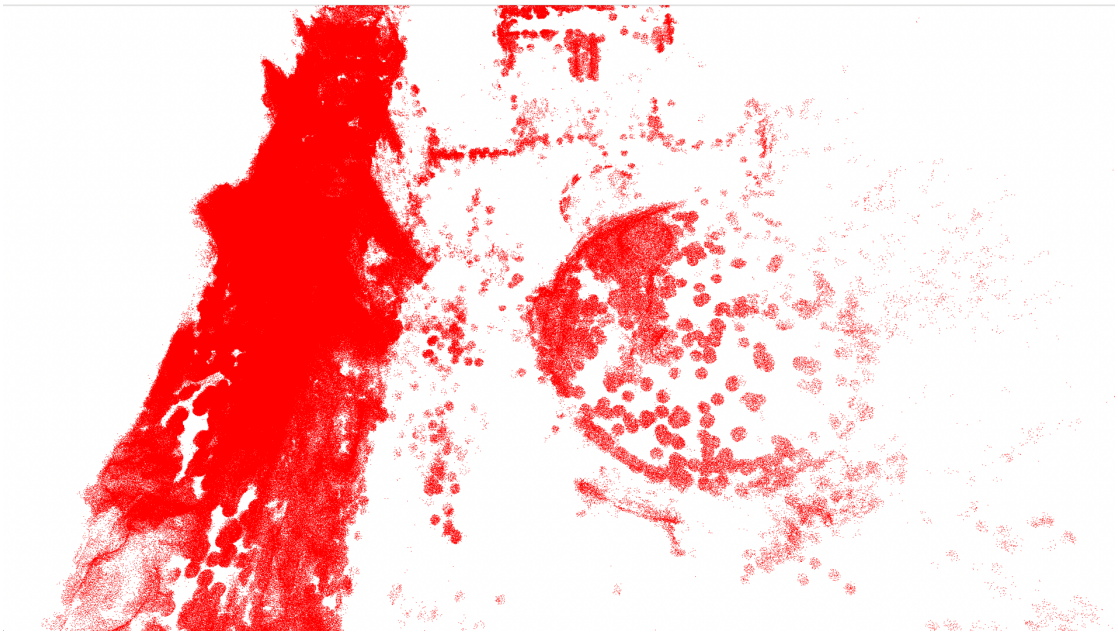

Figure 5.7: Dynamic points segmented on scene P01-01 using Monocular Pipeline.

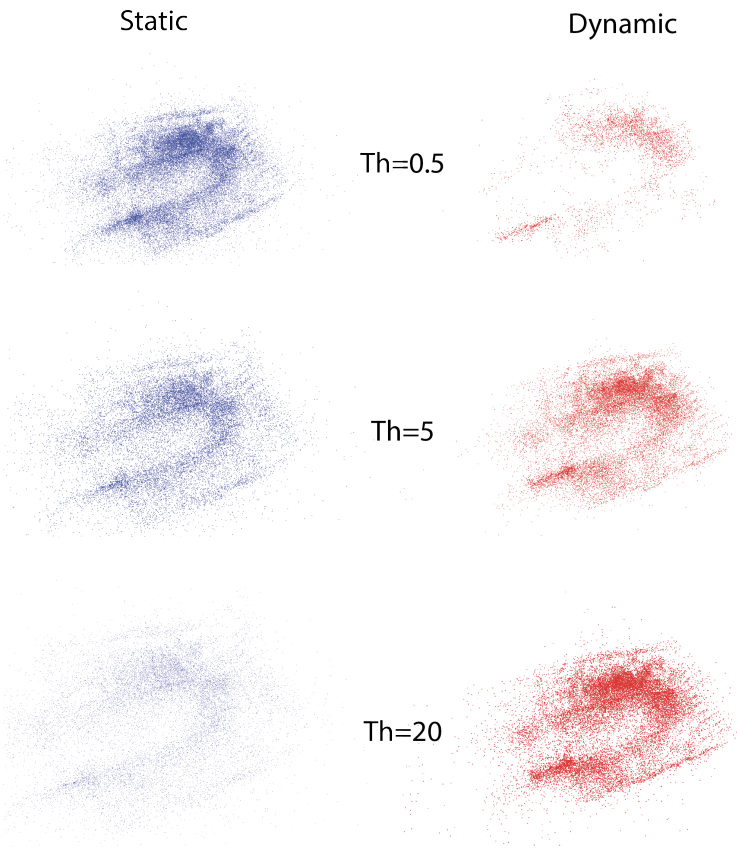

Figure 5.8: Different Segmentation changing the distance threshold. Each point is segmented as dynamic if its distance from a pixel projected in 3D space is less than a threshold Th. The scene is P03-04 and in the left side we can find the static part while on the right side, we have the dynamic points. Here we can notice how this method is pretty inaccurate.

	Original	Sampled	Reconstructed	Obtained	Int/Unif Samples	Threshold
P01-01	98935	5223	4741	3652	**1089**	0.9
P03-04	100251	5060	4522	3654	**868**	0.855
P04-01	69292	4269	3242	2144	**1098**	0.89
P09-02	22187	4953	4398	3495	**903**	0.975
P16-01	74592	5531	5480	4476	**1004**	0.945
P21-01	415853	4655	4588	3733	**855**	0.94

Table 5.3: Split$\sim$ 1000. Number of frames resulting from the different sampling steps. In particular from the original number of frames (Original) are reduced with the homography filter to remove redundancy and keep overlap, resulting in Sampled. The reconstructed frames are the ones which were successfully reconstructed by COLMAP (Reconstructed). The obtained ones are the reconstructed frames filtered again with the homography filter (Obtained). The final samples (Int/Unif Samples) are the reconstructed frames without the previously obtained ones. The thresholds reported are referred to the last homography filter step.

	Original	Sampled	Reconstructed	Obtained	Int/Unif Samples	Threshold
P01-01	98935	5223	4741	4019	**722**	0.91
P03-04	100251	5060	4522	3839	**683**	0.86
P04-01	69292	4269	3242	2463	**779**	0.896
P09-02	22187	4953	4398	3776	**622**	0.977
P16-01	74592	5531	5480	4837	**643**	0.95
P21-01	415853	4655	4588	3942	**646**	0.945

Table 5.4: Split$\sim$ 700.

5.5 Sampling Frames

To work with the *NeuralDiff* pipeline that follows this section, we have to further downsample the frames reconstructed by COLMAP to have some reasonable computational times. We found that at a resolution of 114x64 $\sim$ 1000, a scene took $\sim 4h$ while at 228x128 $\sim$ 700 a scene took $\sim 11h$. In Table 5.4 and Table 5.3 we can see the number of frames kept at each sampling step, obtained using the homography filter and the proposed Intelligent sampling (see Section 4.3). In particular is worth noting the difficulty in obtaining a precise subsample, e.g. all consisting of 700 frames. Each video of each scene has its own characteristics, making it complicated to find a threshold that is valid for all the kitchens. We recall that the threshold sets the minimum value for which frames having overlap over it are removed. So higher value of the threshold means more frames in homography filter, the contrary in Intelligent sampling.

5.6 NeuralDiff Pipeline

Different number of samples for the same scene. To evaluate our final pipeline we started by creating the subsamples upon which we would have trained the neural render. Specifically, we focused on one scene, P01-01, to see how the number of frames selected and the method that selects them affect the pipeline performances. Figure 5.9 gives a visualization of the three different subsampling obtained using the methods presented in Section 4.3 for a total of 217 frames. The left column gives an idea of how the frames are distributed along the video, while in the right side are present the histograms of the samples. We can see that the three method are different from the shapes we obtained. The Intelligent sampling has several peaks of the same size while the others foocus on a specific time instant.

We then proceeded to test the various subsample for P01-01 obtaining the results reported in Table 5.5. As we can see the Intelligent sampling achieves good results. The PSNR is always higher concerning the other methods. It can be seen that all methods suffer the scarcity of frames and in the last subsample, 217, Uniform sampling improves over the Intelligent one by up to 0.09dB. For the static PSNR instead the Intelligent method is always better than the Uniform one, even at low frames. For the mean Average Precision instead, we can see some oscillations, but we have to be careful since our mask is combining the actor and the foreground layer, meaning that the average precision does not assess the ability of the model to distinguish these two parts. For example in Figure 5.10 we can see that in the 400 frames subsamples is exactly present this deficit, where the Uniform sampling is unable to detect the actor even though its mAP is higher than the Intelligent one ($mAP_{Uniform}61.6\% > mAP_{Intelligent}56.63\%$).

Also for our aim to segment dynamic objects, we are interested in the static PSNR (as we obtain dynamic objects as what is NOT static) and Figure 5.10 shows us that the static part is almost identical for each scene.

Qualitative Results Here we present the 3D static reconstruction for P01-01. As shown in Figure 5.11, the first row is the COLMAP pointcloud extracted from the sampled videosequence. Below are placed instead the static reconstructions for the three different sampling strategies. The first thing that comes to our eyes is the overall color which in the Intelligent sampling seems more faithful to the reality. The second thing is the segmentation of the plate on the tabletop, which can be seen in the COLMAP row. The plate is successfully removed in the Intelligent sampling while it is still visible in the other subsamples, although the best model was the Uniform one according to the metrics. Another example is given by the pan highlighted with the green circle which is removed in the Intelligent sampling while not in the others. We can also look at scene P03-04 in Figure 5.12 where the Intelligent sampling manages to remove the can highlighted in red while the

P01-01	Sampling	Durata [s]	PSNR	PSNR statico	mAP
2938	Int.	11 h36 min 59 s	**24.82**	20.41	72.21
2938	Unif	11 h16 min8 s	24.42	20.41	**72.79**
2015	Int.	8 h3 min50 s	**24.51**	**20.46**	**70.49**
2015	Unif	7h 52min 34s	23.93	20.33	69.87
1000	Int.	3h 43min 41s	**23.59**	**20.37**	67.55
1000	Unif	3 h43 min1 s	22.8	20.10	66.51
1000	AU	4 h2 min45 s	23.43	20.31	**67.99**
722	Int.	2h 34 min	**22.65**	**20.20**	**65.42**
722	Unif	2 h30 min7 s	22.09	19.65	62.95
722	AU	2h 36 min46 s	22.48	20.05	64.10
397	Int.	1h 19 min53 s	**21.33**	19.64	56.63
397	Unif	1h 21 min 42 s	20.98	19.54	**61.6**
397	AU	1h 14min 36s	21.2	**19.95**	59.97
217	Int.	40 min55 s	20.32	**19.60**	51.69
217	Unif	41 min8 s	**20.51**	19.42	**53.00**
217	AU	25 min39 s	20.26	19.39	50.58

Table 5.5: NeuralDiff Pipeline Results on P01-01 at 114x64. For the same scene P01-01 results of NeuralDiff pipeline trained on different amount of frames are reported. The Frames are selected using the three different sampling strategis: Intelligent, Uniform and AU (see Section 4.3). The frames are all at a 114x64 resolution.

Uniform sampling can not. Other comparisons for the same scene P01-01 with different frame subsamples are provided in Figure 5.13, where Intelligent sampling manages to reconstruct better colors at lower frames but seems to have similar results at higher ones. In Figure 5.14 we can see a lot of 'noisy' points in the COLMAP reconstruction being removed by our pipeline, leaving the kitchen 'clean'.

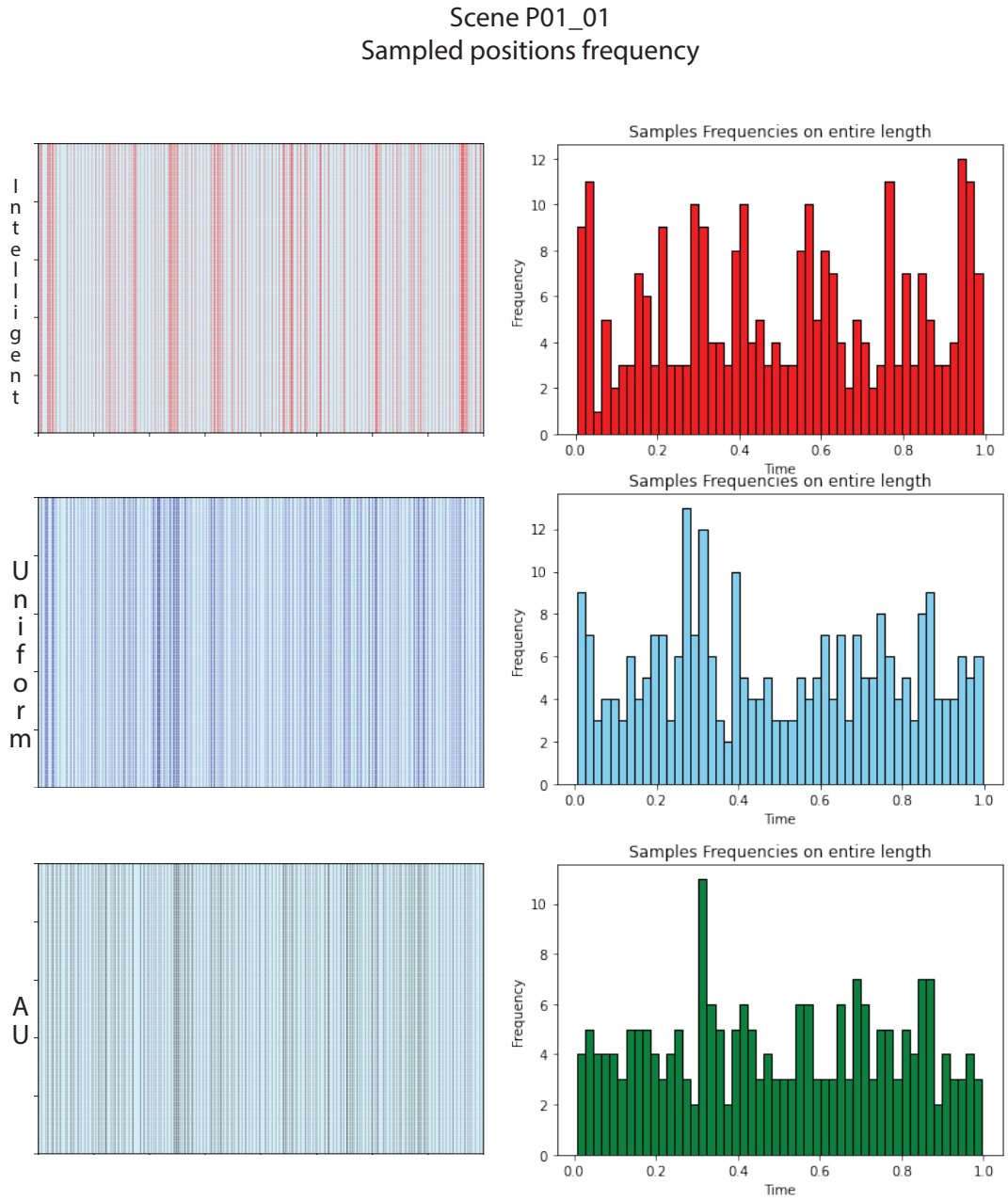

Figure 5.9: Visualization of the sampling of scene P01-01 for the three different methods: Intelligent, Uniform and AU using 217 frames in total. The left box is a proposal we gave to visualize how the frames actually spread along the temporal axis, where a line is drawn in correspondence with each sample. The right boxes represent instead histograms with the frequencies of the samples on the entire duration of the video.

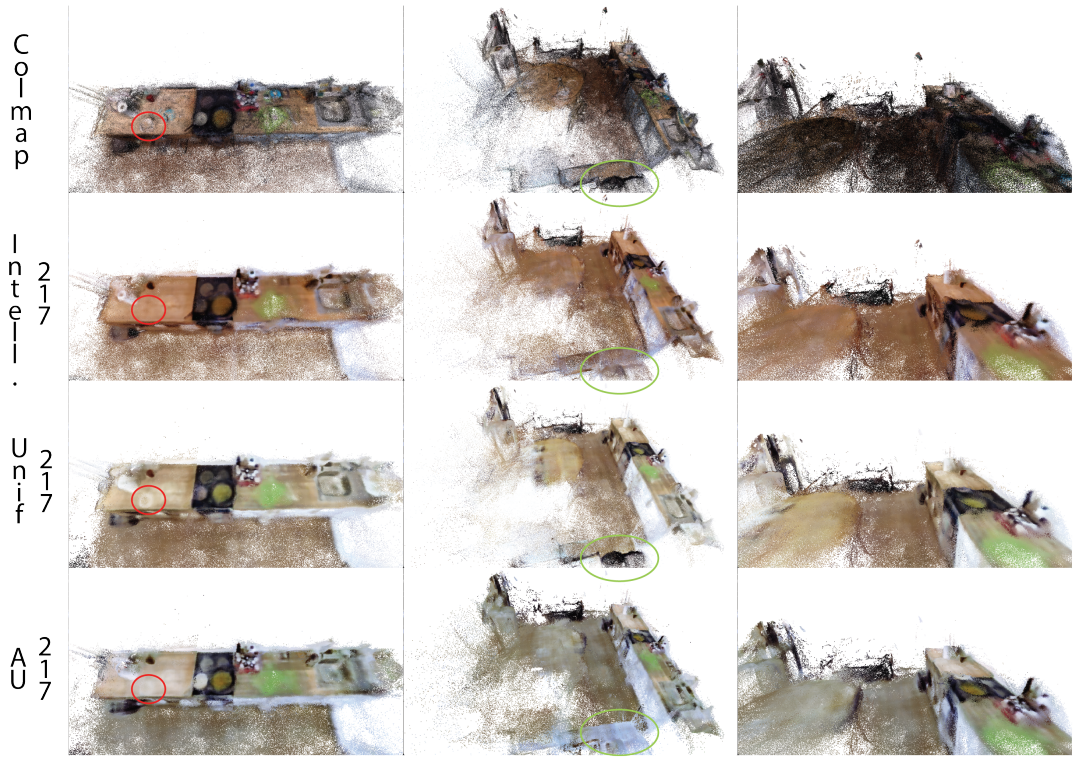

Figure 5.11: Qualitative results for the static reconstruction of P01-01 scene at 217 frames. In red is highlighted a dynamic plate, while in green is a dynamic pan. We can see that Intelligent sampling is correctly removing the objects while Uniform can not.

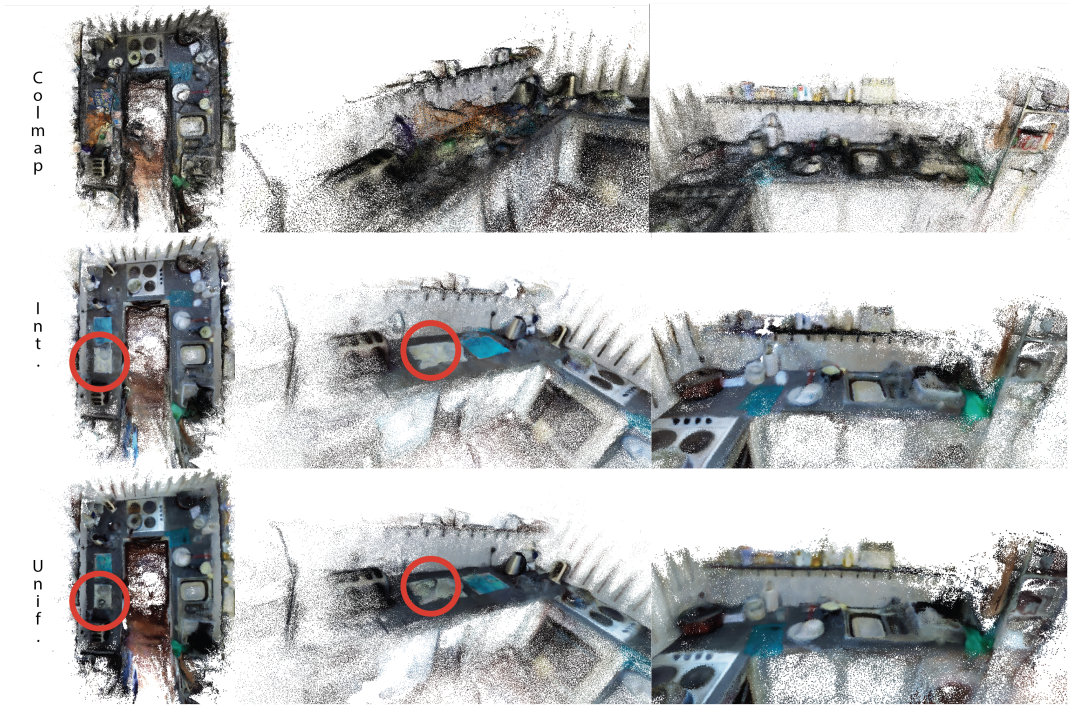

Figure 5.12: Qualitative results for the static reconstruction of P03-04 scene. In red is highlighted a dynamic can that is successfully removed in Intelligent sampling while not in Uniform.

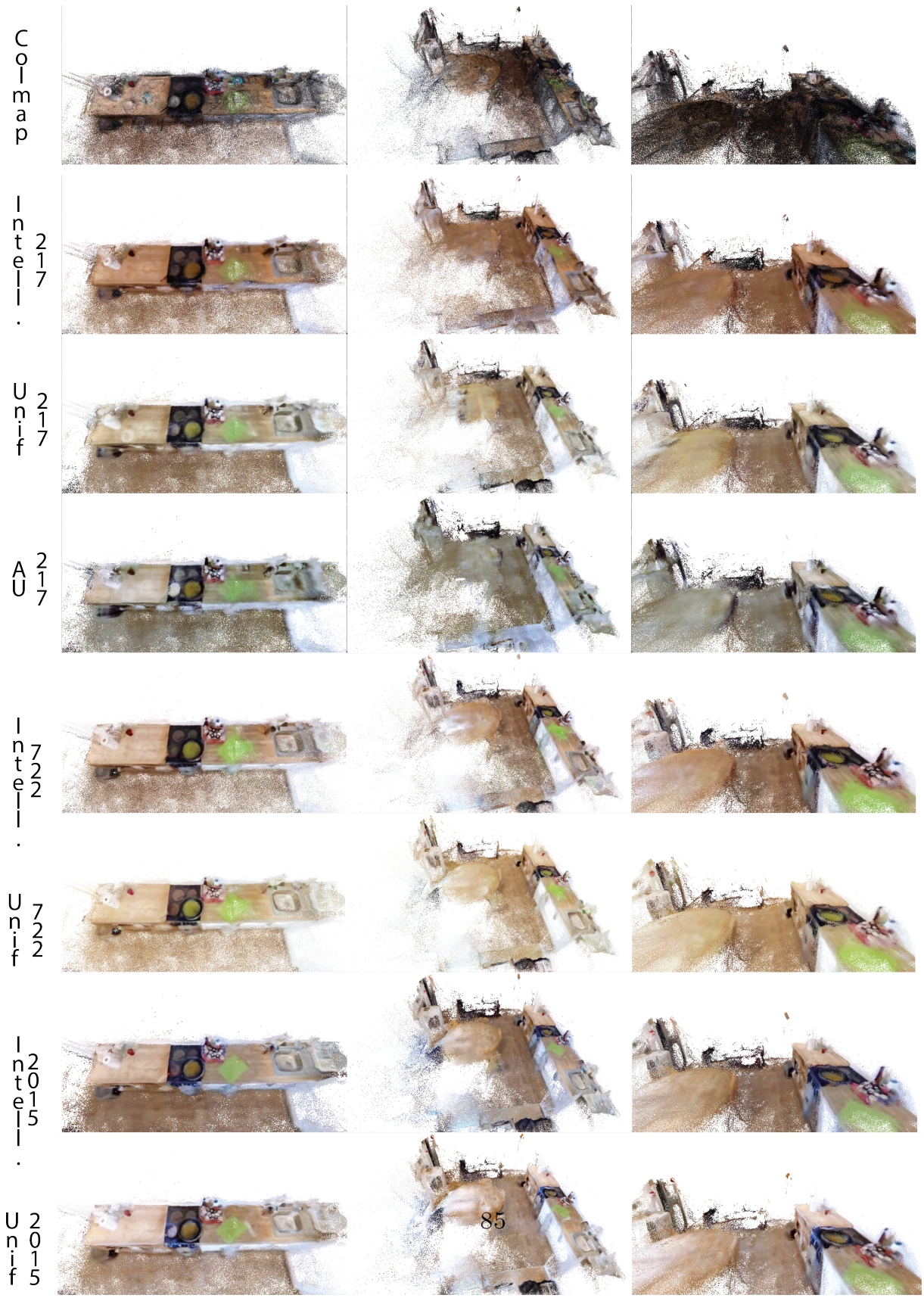

Figure 5.13: Comparative of the qualitative results for different samplings of the P01-01 scene.

Scene	Sampled Frames	Sampling	Durata [s]	PSNR	Improv.	PSNR statico	Improv.	mAP	Improv.
P01-01	1089	Intelligent	3h 43 min41 s	23.59	0.79	20.37	0.27	67.55	1.04
		Uniform	3 h43 min1 s	22.8		20.10		66.51	
P03-04	868	Intelligent	3h 14 min44 s	19.90	0.77	16.73	0	61.92	-2.08
		Uniform	3h 10 min48 s	19.13		16.73		64.00	
P04-01	1098	Intelligent	4h 10 min 12 s	24.32	0.51	21.23	0.74	71.37	5.56
		Uniform	4h 2 min 9s	23.81		20.49		65.81	
P09-02	903	Intelligent	3 h25 min58 s	23.97	0.58	19.43	-0.02	60.05	-1.18
		Uniform	3 h17 min5 s	23.39		19.45		61.23	
P16-01	1004	Intelligent	3h 46 min57 s	22.89	0.14	20.17	0.23	66.89	3.28
		Uniform	3h 43min 11s	22.75		19.94		63.61	
P21-01	855	Intelligent	3h 15 min 59 s	20.02	0.91	15.73	0.66	72.94	4.06
		Uniform	3 h7 min50 s	19.11		15.07		68.88	

Table 5.6: NeuralDiff models trained on different scenes at ∼1000 frames, resolution 114x64. The column Improv. represents the difference between the previous column of the Intelligent split and the Uniform one.

NeuralDiff Pipeline on all scenes. To further validate our results, we repeated the experiments using different scenes using a subsample of ∼1000 frames and one of ∼ 700. The results are reported in Table 5.6 at a resolution of 114x64 and Table 5.7 at a resolution of 228x128. The results show that our method improves, at a resolution of 228x128, on average of +1.96% on the PSNR,+0.54%, and +1.78% on the mAP; while at a resolution of 114x64 we have an improvement of +2.91% on the PSNR,+1.73%, and +1.78% on the mAP.

Scene	Sampled Frames	Sampling	Durata [s]	PSNR	I-U	PSNR statico	I-U	mAP[%]	I-U
P01-01	722	Intelligent	10 h25 min35 s	21.64	0.18	19.58	0.08	66.26	1.93
		Uniform	10h 5 min 7 s	21.46		19.50		64.33	
P03-04	683	Intelligent	9h 40 min5 s	18.89	0.39	16.17	-0.15	61.08	-1.09
		Uniform	9h 16 min59 s	18.5		16.32		62.17	
P04-01	779	Intelligent	10h 45 min 46s	22.15	0.81	20.04	0.46	67.65	4.81
		Uniform	11h 17 min34 s	21.34		19.58		62.84	
P09-02	622	Intelligent	8 h35 min	22.39	0.96	19.10	0.22	67.56	9.34
		Uniform	8 h53 min13 s	21.43		18.88		58.22	
P16-01	643	Intelligent	9h5 min 38s	21.25	0.38	19.38	0.39	63.34	-1.2
		Uniform	9h 7 min 52s	20.87		18.99		64.54	
P21-01	646	Intelligent	9h8 min 54s	18.09	-0.23	15.20	-0.28	65.49	-3.11
		Uniform	9 h8 min 7 s	18.32		15.48		68.60	

Table 5.7: NeuralDiff models trained on different scenes at ∼700 frames, resolution 228x128. The column I-U represents the difference between the previous column of the Intelligent split minus the Uniform one.

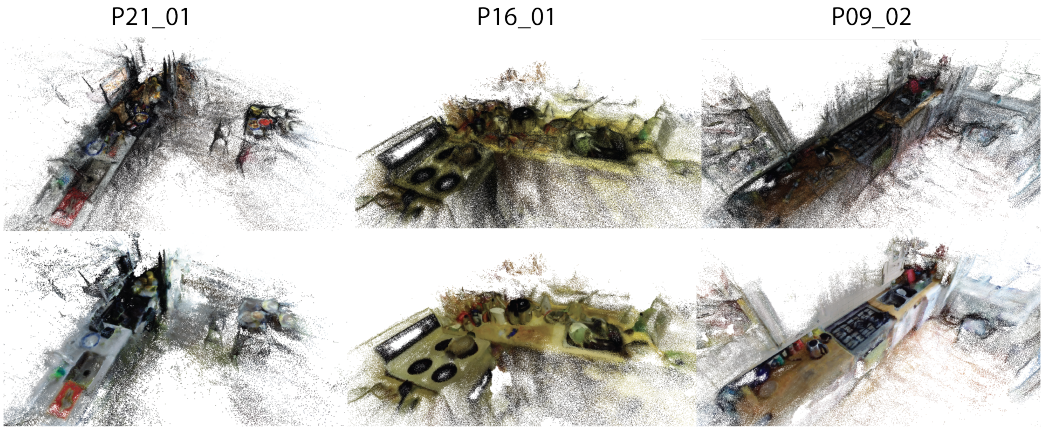

Figure 5.14: Qualitative results of different kitchens. On the first row is reported the COLMAP reconstruction while below is the corresponding cleaned pointcloud.

Sampling and action distributions. Behind the Intelligent sampling which was expressed in Section 4.3, we wanted to validate it. We then verified the existence of a link between the positions of the sampled frames and the frequencies of the objects/actions that were annotated during the videos. In Figure 5.9 the three sampling methods are reported. In Figure 5.15 and Figure 5.16 are reported respectively the comparison of Intelligent and Uniform sampling with the object count, obtained from EPIC-KITCHENS [14] annotations, for scene P01-01 changing the number of samples, as can be read on each sub-figure; and the comparison of Intelligent and Uniform sampling with the object count for each scene with fixed sampling at $\sim$ 1000 frames.

The plots show that the Intelligent sampling technique closely matches the object counts, except for a few scenes. This helps to explain how our method works. It focuses on areas where many actions occur, filtering out lengthy actions and keeping only frames that contain multiple actions. This means that only relevant information is retained.

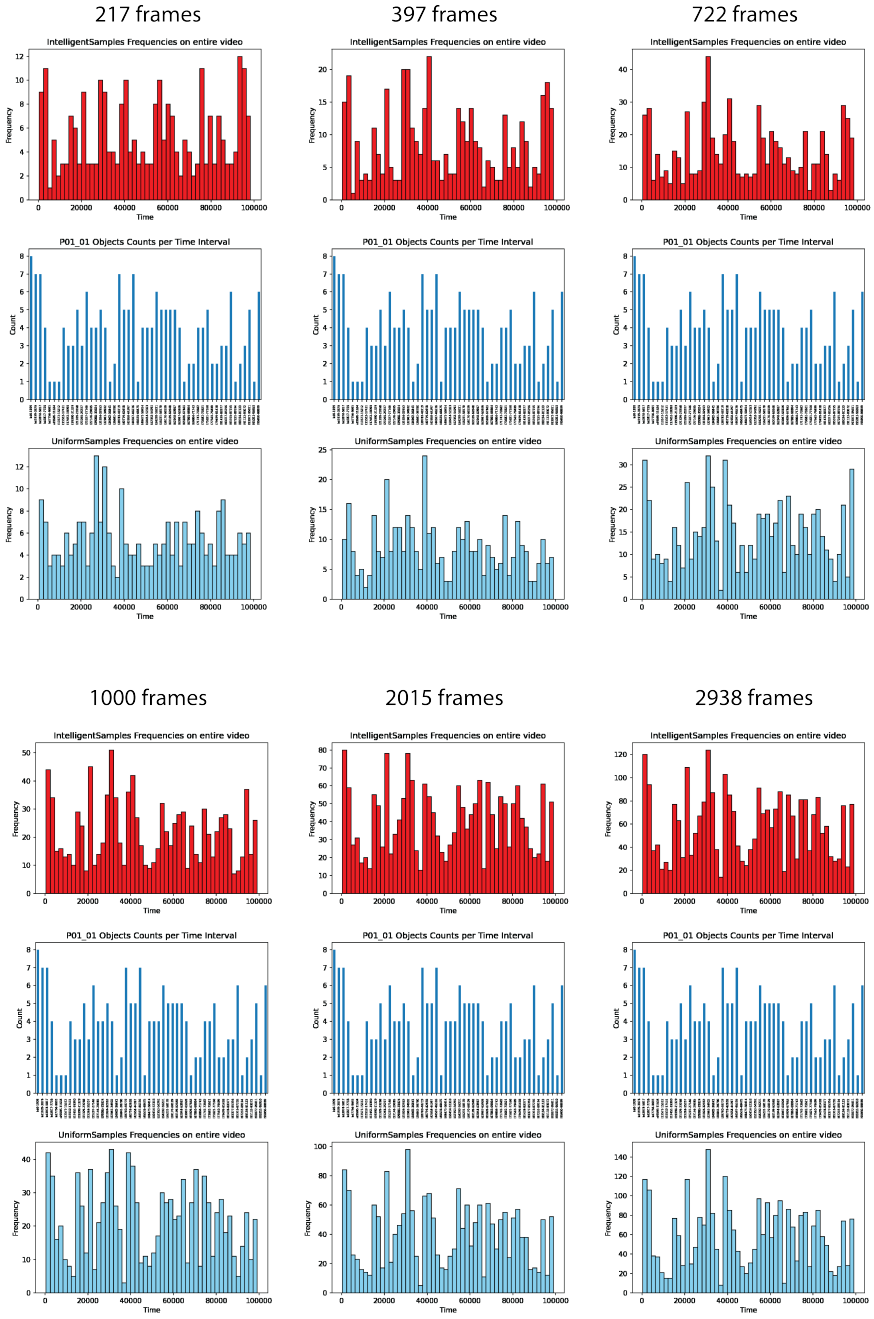

Figure 5.15: Comparison of frequencies for the Intelligent and Uniform sampling with the object count for the P01-01 scene changing the total number of sampled frames.

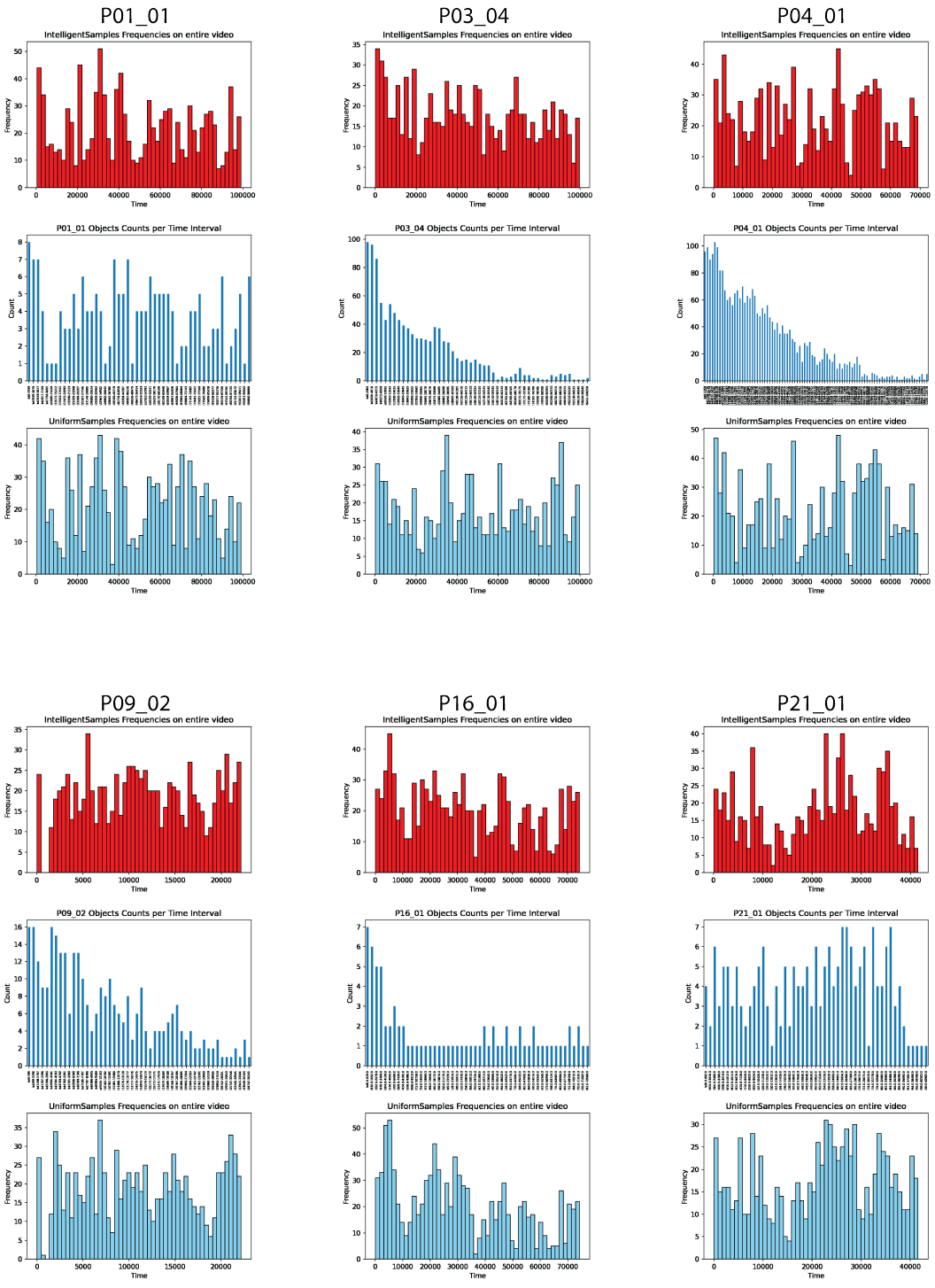

Figure 5.16: Comparison of frequencies for the Intelligent and Uniform sampling with the object count for each scene at a fixed subsample ~ 1000 frames.

Metrics for profile similarity. We also tried to give a quantitative measure of similarity and dissimilarity by comparing some different metrics: Cosine Similarity, Kullback-Leibler Divergence (KLD), Jensen-Shannon Divergence (JSD). More in detail:

- **Cosine Similarity.** It is the cosine of the angle between two vectors. It is derived from the dot product:

$$\mathbf{v_1} \cdot \mathbf{v_2} = \|\mathbf{v_1}\| \, \|\mathbf{v_2}\| \cos(\theta) \tag{5.3}$$

$$CosineSimilarity = \cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{\|\mathbf{v_1}\| \, \|\mathbf{v_2}\|} \tag{5.4}$$

  In our case, we took as vectors the bins of the frames histograms as vector one and the bins of the actions/objects as the second one.

- **Kullback-Leibler Divergence (KLD)** [78] It is a non-symmetric measure of the difference between two probability distributions P and Q. It represents the measure of the lost information when Q is used to approximate P.

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log_2 \left(\frac{P(i)}{Q(i)} \right) \tag{5.5}$$

- **Jensen-Shannon Divergence (JSD).** [79] It is based on the Kullback–Leibler divergence, but it is modified to be symmetric and always has a finite value.

$$\mathrm{JSD}(P\|Q) = \frac{1}{2} D(P\|M) + \frac{1}{2} D(Q\|M) \tag{5.6}$$

  where $M = \frac{1}{2}(P + Q)$ is a mixture distribution of P and Q.

Clearly, we do not have distributions, so we obtained them by normalizing such that their elements summed to 1. The results obtained are reported in Table 5.8, where we can see that the Intelligent sampling performs better than the Uniform sampling both in cosine similarity and in K-L and JS divergences.

5.7 NeuralCleaner

As a last step, after assessing that our method was functioning, we tried to speed up the overall pipeline. We tried to eliminate the actor layer of the *NeuralDiff* pipeline since we are not interested in it, as we only want to know what is moving. The foreground and actor layer are thus merged together and we tried to see how much we could improve by eliminating this distinction. In Table 5.9 we can see the results in comparison with the *NeuralDiff* pipeline. On average we obtained reduced computational cost of $\sim 30\%$.

	Cosine Similarity↑		K-L Div.↓		JS-Div ↓	
	Intelligent	Uniform	Intelligent	Uniform	Intelligent	Uniform
P01-01	**0.9131**	0.9071	**0.1130**	0.1342	**0.0393**	0.0450
P03-04	**0.7638**	0.6773	**0.4224**	0.5389	**0.1609**	0.2025
P04-01	**0.6528**	0.6316	**0.5892**	0.6631	**0.2196**	0.2438
P09-02	**0.7438**	0.7323	2.2962	**1.2859**	**0.1377**	0.1399
P16-01	0.8029	**0.8130**	**0.2170**	0.2431	**0.0727**	0.0815
P21-01	**0.8804**	0.8761	**0.1694**	0.1760	**0.0574**	0.0611

Table 5.8: Metrics comparing the profile of the histograms of frames and annotations. In particular higher values of Cosine similarity indicates similarity; while the value of the two divergences represents the distance between the two distributions.

P01-01	Sampling	Durata Ndiff[s]	Durata [s]	Improvement %	PSNR	PSNR statico	mAP
1089	Int	3h 43min 41s	2h 55 min30 s	-21.54	23.49	19.92	61.00
1089	Unif	3h 43 min1 s	2h 58 min56 s	-19.76	22.86	19.55	56.40
722	Int	2 h34 min	1h 46min 41s	-30.72	22.51	19.47	50.03
722	Unif	2 h30 min7 s	1h 46min 8s	-29.29	22.29	19.46	51.83
397	Int	1h 19min 53s	54 min27 s	-31.83	21.46	19.72	53.69
397	Unif	1h 21min 42s	57 min30 s	-29.62	21.01	19.51	50.60
217	Int	40 min55 s	28 min49 s	-29.57	20.55	18.20	40.87
217	Unif	41 min8 s	29 min 13s	-28.97	20.51	19.45	46.61

Table 5.9: Results for NeuralCleaner, compared with the durations of the NeuralDiff pipeline. We can see that the durations are on average shorter of ∼30%.

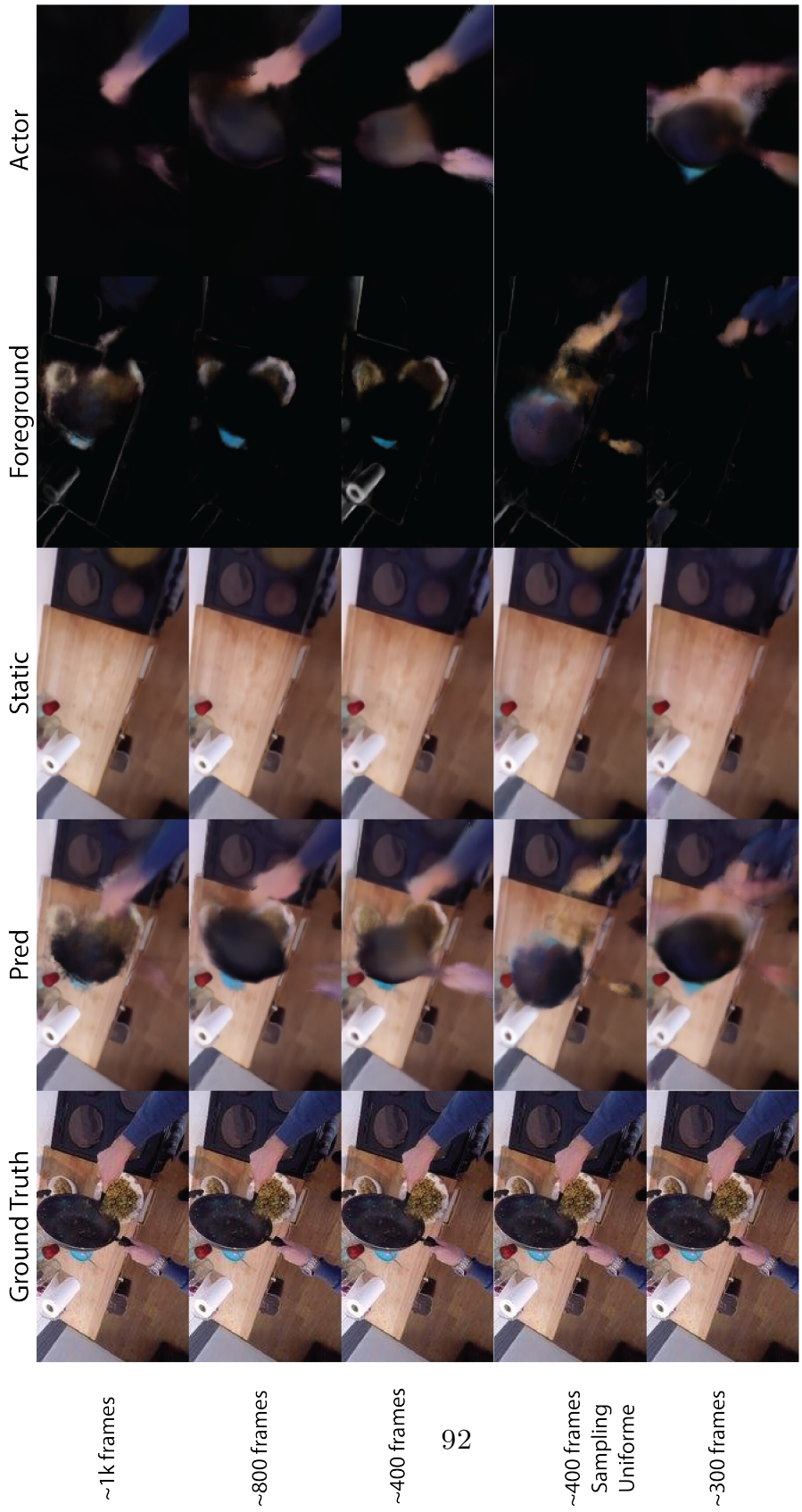

Figure 5.10: Qualitative results on P01-01 at 228x128 Visualization of the output of the different models trained on different subsamplings for scene P01-01. The first column represent the real frame while the next ones are respectively: the predicted image, which is the combination of: the static part, the foreground, and the actor part.

Part III

Conclusions

Chapter 6

Conclusions

In conclusion, we have explored the new spreading *egocentric videos* trends and how the research's attention was captured by these new data. Specifically, we have covered *Neural Rendering*, a technique that employs neural networks to learn 3D representation of a scene, and *Photogrammetry* which is the study on how to obtain information about a scene starting from some gathered images depicting it. We have also analyzed the new unique challenges introduced by egocentric data, such as motion blur, occlusion, restricted field of view, etc., as well as strategies to tackle them.

We started by using EPIC-KITCHENS [14], one of the largest egocentric datasets containing video recordings of cooking routines of different peoples from different countries in kitchen scenarios, annotated with actions and object labels. We proceeded to test and evaluate on it a Structure from Motion algorithm, COLMAP [15], and we assessed its limitation on these types of data among which it stands out fuzzy reconstruction.

Further on, we proposed two pipelines to compensate for the COLMAP weaknesses, by segmenting dynamic objects in the 3D reconstruction point cloud and removing them in a second stage. The pipelines share the same basic building blocks: COLMAP to retrieve the camera poses and *NeuralDiff* to segment moving objects in 2D. These two pipelines differ in the presence of just one element: *Monocular Depth Estimator*, a pre-trained neural network that has learned to predict the frames' pixels depth. If also camera position and direction are available we could project each pixel in the 3D world using projective geometry. The first pipeline, which we have called *Monocular Pipeline*, bases its functioning on this element. After the NeuralDiff block has decomposed the frame in the static and moving layers, the latter are projected in the 3D reconstruction point cloud. The distance of the projected points from the reconstruction ones will then determine if the reconstructed point is dynamic or not. The second pipeline, which we refer to as *NeuralDiff Pipeline*, on the other hand, lacks the presence of the *Monocular*

Depth Estimator and exploits the intrinsic knowledge of the 3D scene learned by the *NeuralDiff* block. In fact, during training, this block learns to give color and opacity to any queried 3D point which is then averaged with other points to be rendered on a 2D surface. We modified it such that we are no longer interested in rendering an image but just in querying 3D points. In this way, after each neural renderer is trained we can query points to the static stream and segment them as dynamic if their opacity is under a certain threshold.

The results clearly stated that *NeuralDiff Pipeline* outperformed *Monocular Pipeline*. The latter was heavily dependent on the scene to decide a segmenting threshold and also the projection of the frames in the 3D world was not as accurate as we expected.

We assessed the robustness of the *NeuralDiff* pipeline by testing different parameters such as the total number of frames used, the resolution of the frames, and the type of sampling from which the frames were derived. We proposed a new sampling method aimed at removing any redundant frames and reducing their total number, such as to speed up the training, by minimizing the overlap between frames. This technique revealed itself to be successful as at an equal number of frames our method gave better results than the basic one based on uniform extraction. With these experiments, we also found a practical explanation of the reason why our sampling method performed better and it is linked to the action/object annotations frequency, which showed a similar pattern along the duration of the video.

As a last step, we modified *NeuralDiff* to better fulfill our goals, removing the distinction between actor and foreground during layer discrimination allowed us to remove a neural network block resulting in lighter training, obtaining what we called *NeuralCleaner Pipeline*.

Further research on this topic is necessary as it presents multiple unexplored paths. A promising one is the freshly introduced *Gaussian Splatting*, a rendering algorithm that outperformed neural radiance field capable of obtaining real-time results, as opposed to the ∼hours required by NeRF.

Bibliography

[1] Thi-Hoa-Cuc Nguyen, Jean-Christophe Nebel, and Francisco Florez-Revuelta. «Recognition of Activities of Daily Living with Egocentric Vision: A Review». In: *Sensors* 16.1 (2016). ISSN: 1424-8220. DOI: 10.3390/s16010072. URL: https://www.mdpi.com/1424-8220/16/1/72 (cit. on p. 1).

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV] (cit. on p. 1).

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. arXiv: 1606.00915 [cs.CV] (cit. on p. 1).

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. «Simple online and realtime tracking». In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016. DOI: 10.1109/icip.2016.7533003. URL: http://dx.doi.org/10.1109/ICIP.2016.7533003 (cit. on p. 1).

[5] Chiara Plizzari, Gabriele Goletto, Antonino Furnari, Siddhant Bansal, Francesco Ragusa, Giovanni Maria Farinella, Dima Damen, and Tatiana Tommasi. *An Outlook into the Future of Egocentric Vision*. 2024. arXiv: 2308.07123 [cs.CV] (cit. on p. 1).

[6] H Fuchs, DV Ginzel, G Paar, and M Suppa. «Stereo Vision for Planetary Rovers – Staying on Track». In: *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*. 2003 (cit. on p. 2).

[7] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. «Real-Time Dense Reconstruction of Laparoscopic Scenes». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2011 (cit. on p. 2).

[8] Marco Callieri, Paolo Cignoni, Fabio Ganovelli, Claudio Montani, Paolo Pingi, Federico Ponchio, and Roberto Scopigno. «Digital reconstruction and the renaissance museum: a case study at the museo di san marco in florence». In: *Proceedings of the 2008 14th International Conference on Virtual Systems and Multimedia*. IEEE. 2008 (cit. on p. 2).

[9] Thierry Bouwmans. «Traditional and recent approaches in background modeling for foreground detection: An overview». In: *Computer Science Review* 11-12 (2014), pp. 31–66. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2014.04.001. URL: https://www.sciencedirect.com/science/article/pii/S1574013714000033 (cit. on pp. 2, 23).

[10] Jana Mattheus, Hans Grobler, and Adnan M. Abu-Mahfouz. «A Review of Motion Segmentation: Approaches and Major Challenges». In: *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. 2020, pp. 1–8. DOI: 10.1109/IMITEC50163.2020.9334076 (cit. on pp. 2, 23, 32).

[11] Vadim Tschernezki, Diane Larlus, and Andrea Vedaldi. «NeuralDiff: Segmenting 3D objects that move in egocentric videos». In: *CoRR* abs/2110.09936 (2021). arXiv: 2110.09936. URL: https://arxiv.org/abs/2110.09936 (cit. on pp. 2–4, 37, 53, 54, 61, 62, 64, 69, 70).

[12] Johannes Lutz Schönberger and Jan-Michael Frahm. «Structure-from-Motion Revisited». In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 3, 4, 17).

[13] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. «Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.3 (2022) (cit. on pp. 3, 20, 21).

[14] Dima Damen et al. «Scaling Egocentric Vision: The EPIC-KITCHENS Dataset». In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on pp. 3, 36, 40, 42–46, 87, 94).

[15] Colmap 2024. URL: https://colmap.github.io/index.html (cit. on pp. 3, 4, 16, 33, 36, 62, 94).

[16] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. «Vision Transformers for Dense Prediction». In: *ICCV* (2021) (cit. on pp. 3, 4, 19, 20, 63).

[17] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV] (cit. on pp. 3, 24, 28, 32, 37, 53).

[18] Peter Sturm. «Pinhole Camera Model». In: *Computer Vision: A Reference Guide*. Ed. by Katsushi Ikeuchi. Boston, MA: Springer US, 2014, pp. 610–613. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_472. URL: https://doi.org/10.1007/978-0-387-31439-6_472 (cit. on p. 7).

[19] Chatz Schwab. *The Evolution of Photography: From Camera Obscura to Digital Cameras - A Timeline*. Dec. 2021. DOI: 10.5281/zenodo.5795101. URL: https://doi.org/10.5281/zenodo.5795101 (cit. on p. 7).

[20] OpenCV 2024. URL: https://docs.opencv.org/4.x/df/d54/tutorial_py_features_meaning.html (cit. on p. 11).

[21] David G. Lowe. «Distinctive Image Features from Scale-Invariant Keypoints». In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94. URL: http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf (cit. on p. 11).

[22] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. «Speeded-Up Robust Features (SURF)». In: *Computer Vision and Image Understanding* 110.3 (2008). Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555 (cit. on p. 11).

[23] Edward Rosten and Tom Drummond. «Machine learning for high-speed corner detection». In: *European Conference on Computer Vision*. Springer. 2006 (cit. on p. 11).

[24] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. «BRIEF: Binary Robust Independent Elementary Features». In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15561-1 (cit. on p. 12).

[25] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. «ORB: An efficient alternative to SIFT or SURF». In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544 (cit. on p. 12).

[26] Marius Muja and David G Lowe. «FLANN: Fast Library for Approximate Nearest Neighbors». In: *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*. 2009 (cit. on p. 12).

[27] Andrew W. Fitzgibbon, Simon W. Bowman, Andrew Zisserman, and Mark A. Fischler. «Bundle Adjustment – A Modern Synthesis». In: *Vision Algorithms: Theory and Practice*. Vol. 1883. LNCS. Springer-Verlag, 2000, pp. 298–372 (cit. on p. 16).

[28] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. «Pixelwise View Selection for Unstructured Multi-View Stereo». In: *European Conference on Computer Vision (ECCV)*. 2016 (cit. on p. 17).

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL] (cit. on p. 19).

[30] Marc Levoy and Pat Hanrahan. «Light field rendering». In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. SIGGRAPH96. ACM, Aug. 1996. DOI: 10.1145/237170.237199. URL: http://dx.doi.org/10.1145/237170.237199 (cit. on p. 22).

[31] Steven Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael Cohen. «The Lumigraph». In: *Proc. of SIGGRAPH 96* 96 (Aug. 2001). DOI: 10.1145/237170.237200 (cit. on p. 22).

[32] Michael Waechter, Nils Moehrle, and Michael Goesele. «Let There Be Color! Large-Scale Texturing of 3D Reconstructions». In: *European Conference on Computer Vision*. 2014. URL: https://api.semanticscholar.org/CorpusID:6085476 (cit. on p. 22).

[33] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. «Unstructured lumigraph rendering». In: *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001). URL: https://api.semanticscholar.org/CorpusID:215780580 (cit. on p. 22).

[34] Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. *Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer*. 2019. arXiv: 1908.01210 [cs.CV] (cit. on p. 22).

[35] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. *Unsupervised Training for 3D Morphable Model Regression*. 2018. arXiv: 1806.06098 [cs.CV] (cit. on p. 22).

[36] K.N. Kutulakos and S.M. Seitz. «A theory of shape by space carving». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. 1999, 307–314 vol.1. DOI: 10.1109/ICCV.1999.791235 (cit. on p. 23).

[37] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. *DeepView: View Synthesis with Learned Gradient Descent*. 2019. arXiv: 1906.07316 [cs.CV] (cit. on p. 23).

[38] Philipp Henzler, Volker Rasche, Timo Ropinski, and Tobias Ritschel. *Single-image Tomography: 3D Volumes from 2D Cranial X-Rays*. 2018. arXiv: 1710.04867 [cs.GR] (cit. on p. 23).

[39] Abhishek Kar, Christian Häne, and Jitendra Malik. «Learning a Multi-View Stereo Machine». In: *ArXiv* abs/1708.05375 (2017). URL: https://api.semanticscholar.org/CorpusID:19285959 (cit. on p. 23).

[40] Pia Bideau and Erik Learned-Miller. *It's Moving! A Probabilistic Model for Causal Motion Segmentation in Moving Camera Videos*. 2016. arXiv: 1604.00136 [cs.CV] (cit. on p. 23).

[41] Thomas Brox and Jitendra Malik. «Object Segmentation by Long Term Analysis of Point Trajectories». In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 282–295. ISBN: 978-3-642-15555-0 (cit. on p. 23).

[42] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. *Self-supervised Video Object Segmentation by Motion Grouping*. 2021. arXiv: 2104.07658 [cs.CV] (cit. on p. 23).

[43] James T. Kajiya and Brian Von Herzen. «Ray tracing volume densities». In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984). URL: https://api.semanticscholar.org/CorpusID:6722621 (cit. on p. 25).

[44] N. Max. «Optical models for direct volume rendering». In: *IEEE Transactions on Visualization and Computer Graphics* 1.2 (1995), pp. 99–108. DOI: 10.1109/2945.468400 (cit. on p. 26).

[45] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. *On the Spectral Bias of Neural Networks*. 2019. arXiv: 1806.08734 [stat.ML] (cit. on p. 27).

[46] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. *Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines*. 2019. arXiv: 1905.00889 [cs.CV] (cit. on pp. 28, 30).

[47] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. *Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations*. 2020. arXiv: 1906.01618 [cs.CV] (cit. on pp. 28, 30).

[48] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. *NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections*. 2021. arXiv: 2008.02268 [cs.CV] (cit. on pp. 32, 36, 37).

[49] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. *Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes*. 2021. arXiv: 2011.13084 [cs.CV] (cit. on p. 33).

[50] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. *Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video*. 2021. arXiv: 2012.12247 [cs.CV] (cit. on p. 33).

[51] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. *NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections*. 2021. arXiv: 2008.02268 [cs.CV] (cit. on pp. 35, 53).

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (cit. on p. 40).

[53] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497 (cit. on p. 40).

[54] Andrej Karpathy and Li Fei-Fei. «Deep Visual-Semantic Alignments for Generating Image Descriptions». In: *CoRR* abs/1412.2306 (2014). arXiv: 1412.2306. URL: http://arxiv.org/abs/1412.2306 (cit. on p. 40).

[55] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. «VQA: Visual Question Answering». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on p. 40).

[56] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. «The Pascal Visual Object Classes (VOC) Challenge.» In: *Int. J. Comput. Vis.* 88.2 (2010), pp. 303–338. URL: http://dblp.uni-trier.de/db/journals/ijcv/ijcv88.html#EveringhamGWWZ10 (cit. on p. 40).

[57] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 40).

[58] Tsung-Yi Lin et al. «Microsoft COCO: Common Objects in Context». In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312 (cit. on p. 40).

[59] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. «Scene Parsing through ADE20K Dataset». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5122–5130. DOI: 10.1109/CVPR.2017.544 (cit. on p. 40).

[60] Raghav Goyal et al. «The "something something" video database for learning and evaluating visual common sense». In: *CoRR* abs/1706.04261 (2017). arXiv: 1706.04261. URL: http://arxiv.org/abs/1706.04261 (cit. on p. 40).

[61] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. «YouTube-8M: A Large-Scale Video Classification Benchmark». In: *CoRR* abs/1609.08675 (2016). arXiv: 1609.08675. URL: http://arxiv.org/abs/1609.08675 (cit. on p. 40).

[62] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. «A Dataset for Movie Description». In: *CoRR* abs/1501.02530 (2015). arXiv: 1501.02530. URL: http://arxiv.org/abs/1501.02530 (cit. on p. 40).

[63] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. «MovieQA: Understanding Stories in Movies through Question-Answering». In: *CoRR* abs/1512.02902 (2015). arXiv: 1512.02902. URL: http://arxiv.org/abs/1512.02902 (cit. on p. 40).

[64] David F. Fouhey, Weicheng Kuo, Alexei A. Efros, and Jitendra Malik. «From Lifestyle Vlogs to Everyday Interactions». In: *CoRR* abs/1712.02310 (2017). arXiv: 1712.02310. URL: http://arxiv.org/abs/1712.02310 (cit. on p. 40).

[65] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. «Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding». In: *CoRR* abs/1604.01753 (2016). arXiv: 1604.01753. URL: http://arxiv.org/abs/1604.01753 (cit. on p. 40).

[66] Dima Damen et al. «Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100». In: *International Journal of Computer Vision (IJCV)* 130 (2022), pp. 33–55. URL: https://doi.org/10.1007/s11263-021-01531-2 (cit. on pp. 48, 49, 51).

[67] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL] (cit. on p. 50).

[68] Vadim Tschernezki, Ahmad Darkhalil, Zhifan Zhu, David Fouhey, Iro Laina, Diane Larlus, Dima Damen, and Andrea Vedaldi. *EPIC Fields: Marrying 3D Geometry and Video Understanding*. 2024. arXiv: 2306.08731 [cs.CV] (cit. on pp. 50, 52, 54–56, 67, 72).

[69] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F. Fouhey. *Understanding Human Hands in Contact at Internet Scale*. 2020. arXiv: 2006.06669 [cs.CV] (cit. on p. 51).

[70] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. *Large-scale interactive object segmentation with human annotators*. 2019. arXiv: 1903.10830 [cs.CV] (cit. on p. 51).

[71] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. *Monocular Dynamic View Synthesis: A Reality Check*. 2022. arXiv: 2210.13445 [cs.CV] (cit. on p. 54).

[72] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. *Self-supervised Video Object Segmentation by Motion Grouping*. 2021. arXiv: 2104.07658 [cs.CV] (cit. on p. 55).

[73] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. *EPIC-KITCHENS VISOR Benchmark: VIdeo Segmentations and Object Relations*. 2022. arXiv: 2209.13064 [cs.CV] (cit. on pp. 56, 57).

[74] Kristen Grauman et al. *Ego4D: Around the World in 3,000 Hours of Egocentric Video*. 2022. arXiv: 2110.07058 [cs.CV] (cit. on p. 57).

[75] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Carl Yuheng Ren. *Aria Digital Twin: A New Benchmark Dataset for Egocentric 3D Machine Perception*. 2023. arXiv: 2306.06362 [cs.CV] (cit. on p. 58).

[76] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. «A Signal Fidelity Criterion for Image Quality: An Overview of MSE and PSNR». In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612 (cit. on p. 70).

[77] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. «The PASCAL Visual Object Classes (VOC) Challenge». In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338 (cit. on p. 71).

[78] Solomon Kullback and Richard A Leibler. «On information and sufficiency». In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86 (cit. on p. 90).

[79] Junlin Lin. «A study in comparing divergence measures for binary data». In: *Divergence Measures Based on the Shannon Entropy* (1991) (cit. on p. 90).