POLITECNICO DI TORINO

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Segmenting dynamic points in 3D scenarios

Supervisors

Prof. Tatiana TOMMASI

Prof. Chiara PLIZZARI

Candidate

Francesco BORGNA

March 2024

Summary

Ma che dici Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Acknowledgements

ACKNOWLEDGMENTS

"HI"
Goofy, Google by Google

Table of Contents

List of Tables

List of Figures

Acronyms

SfM
: Structure from Motion

pcd
: Pointcloud

Part I

Related Works

1. Epic Kitchens
2. Epic Fields
3. Photogrammetry
4. COLMAP
5. NeRF
6. NeuralDiff
7. Monocular Depth Estimation
8. motion estimation
9. (N3F)
10. (Gaussian Splatting)

Chapter 1

Datasets

Motivazioni per cui abbiamo usato questi datasets? The choice of the following datasets was dictated by the fact that up to our knowledge these are currently the largest datasets available in the egocentric scenarios. Let us see these in details.

1.1 EPIC-Kitchens

EPIC-Kitchens [1] 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 non scripted actions, meaning that the actors were not following any *scripted* actions(we will see this in more detail later).

1.1.1 Introduction/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 have been seen in many domains such as image classification [2],object detection [3], captioning [4] and visual question answering [5]; due to the advances in deep learning but mainly due to the availability of large-scale image benchmarks such as PASCAL VOC [6],ImageNet [7],Microsoft COCO [8], ADE20K [9]. In the same way the authors thought that by introducing a large scale video dataset could contribute to the development of video domains.

Some video datasets were already available for action classification [10, 11, 12, 13, 14] but, a part from [13], these all contain very short videos, focusing on just a single action. A solution to this problem was given by Charades [15] where 10k videos have been collected of humans performing daily tasks at home. The problem with this dataset is that the action 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 actions, increasing the possibilities of a succesful 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 1.1 we report a summary of the datasets compared by the authors.

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

1.1.2 Data Collection

To the data collection were 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 visit the kitchen for three consecutive days, starting filming just before entering the kitchen and stopping before leaving it. They participated to the process of their own free will without being paid in any way.

Few requests were asked to 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 at the middle of the camera frame. The camera settings was set for most of videos to linear field of view, using 59.94fps as frame rate and Full HD resolution 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.

1.1.3 Data Annotation pipeline

1.2 (EPIC-Fields)

?

1.3 ((Ego4D))

?

Chapter 2

Neural Rendering

Chapter 3

Photogrammetry

Photogrammetry is the science and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring and interpreting photographic images and patterns of electromagnetic radiant imagery and other phenomena[16].

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

As first assumption we have the modellization of the camera as a simplified version of itself: the *Pinhole Camera*. As in the first designed cameras(*camera obscura*), in the Pinhole Camera world's light is captured through a pinhole and then projected into the *focal plane*.

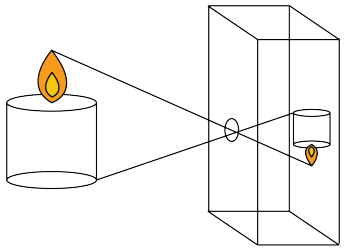

Figure 3.1: Pinhole Camera

A Pinhole camera is characterized by two collection of parameters:

- **Extrinsic** parameters: gives us information on location and rotation in the world.

- **Intrinscic** parameters: gives us internal property such as: focal length, field of view, resolution...

These parameters can be rewritten in their corresponding matrices:

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

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).

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

where:

- $\mathbf{R}_{3x3}$ is a rotation matrix (Aggiungi le tre combinazioni...)
- $\mathbf{t}_{3x1}$ is a translation vector

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

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

$$\begin{bmatrix} u \\ v \\ z \end{bmatrix} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{t}_{3x1} \\ 0_{1x3} & \mathbf{1}_{1x1} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

3.1 Structure from Motion: SfM

Structure from motion (SfM) is the process of estimating the 3-D structure of a scene from a set of 2-D images. SfM is used in many applications, such as 3-D scanning, augmented reality, and visual simultaneous localization and mapping [17].

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

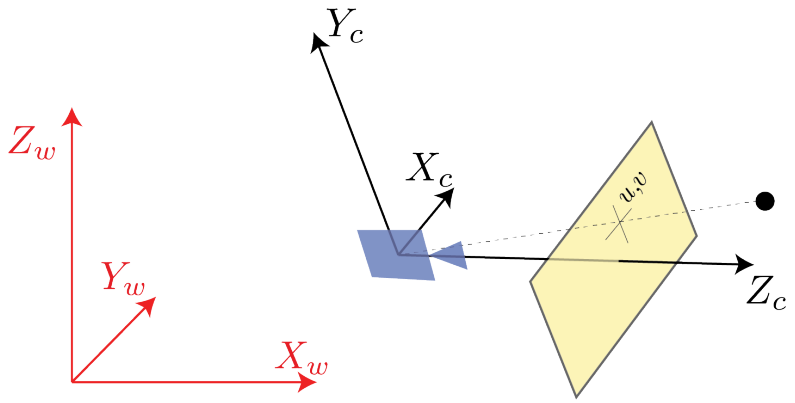

Figure 3.2: Reference systems

different methods. Also the **ordering** of the frames may be relevant for a succesful reconstruction.

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

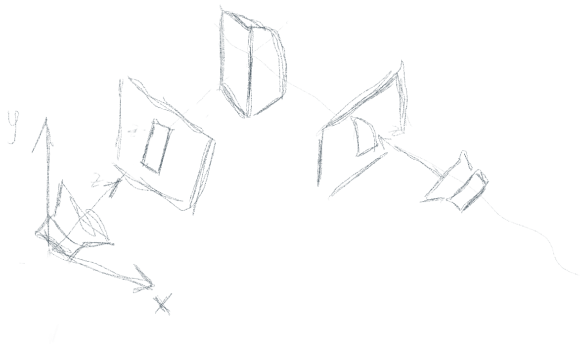

Figure 3.3: Basic SfM Scenario

In this case the 3-D structure can be recovered *up to scale*, meaning that the relations between the obtained points is 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 informations from other sensors.

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

1. **Feature Detection** and **Matching**

2. Estimating **Fundamental matrix**

3. Estimating **Essential Matrix** from Fundamental matrix

4. Estimating **Camera Pose** from Essential Matrix

3.1.1 Feature Detection and Matching

In order 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 computational inefficient and possibly misleading due to color, it has been shown succesful to use *feature points*. Feature points are relavant points which encapsulate the local appeareance 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 us humans find correspondances in pictures, looking at image 3.4. 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 correspondances, as happens instead for patches A or B.

Figure 3.4: Feature Detection Example [18]

In the same way feature extractor works for computers. Two of the first algorithms are infact based on corners 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 report some of them as reported in [18]:

- **SIFT** [19]: Harris corner detector is not good enough when scale of image changes. Lowe developed a breakthrough method to find scale-invariant features and it is called SIFT
- **SURF** (Speeded-Up Robust Features) [20]: faster SIFT version
- **FAST Algorithm for corner detection** All the above feature detection methods are good in some way. But they are not fast enough to work in real-time applications like SLAM. There comes the FAST algorithm, which is really "FAST".
- **BRIEF**(Binary Robust Independent Elementary Features) [21]:SIFT uses a feature descriptor with 128 floating point numbers. Consider thousands of

such features. It takes lots of memory and more time for matching. We can compress it to make it faster. But still we have to calculate it first. There comes BRIEF which gives the shortcut to find binary descriptors with less memory, faster matching, still higher recognition rate.

- **ORB(Oriented FAST and Rotated BRIEF)** [22]: Open source alternative to SIFT and SURF released by OpenCV.

Figure 3.5: SIFT features extracted

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

- **Brute-Force Matcher**: It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.
- **FLANN**: 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. It works faster than BFMatcher for large datasets.

3.1.2 Estimating Fundamental and Essential Matrices

The Fundamental (F) and the Essential (E) matrices allow 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(3.7), given two cameras C_l and C_r the following definitions can be given:

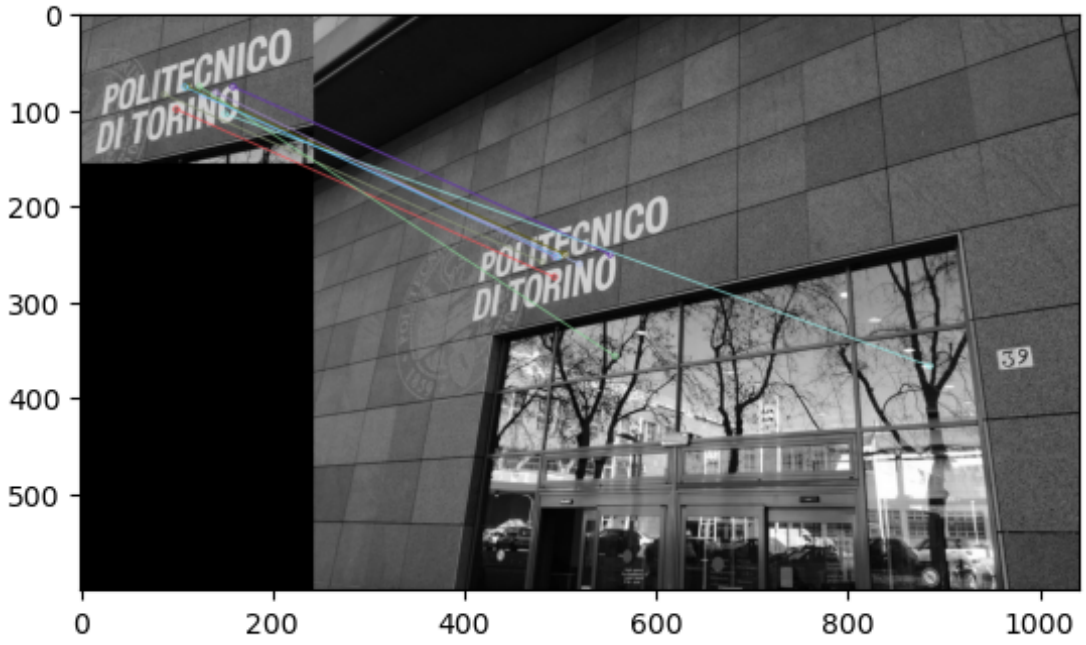

Figure 3.6: Features correspondence computed using BruteForce Matcher.

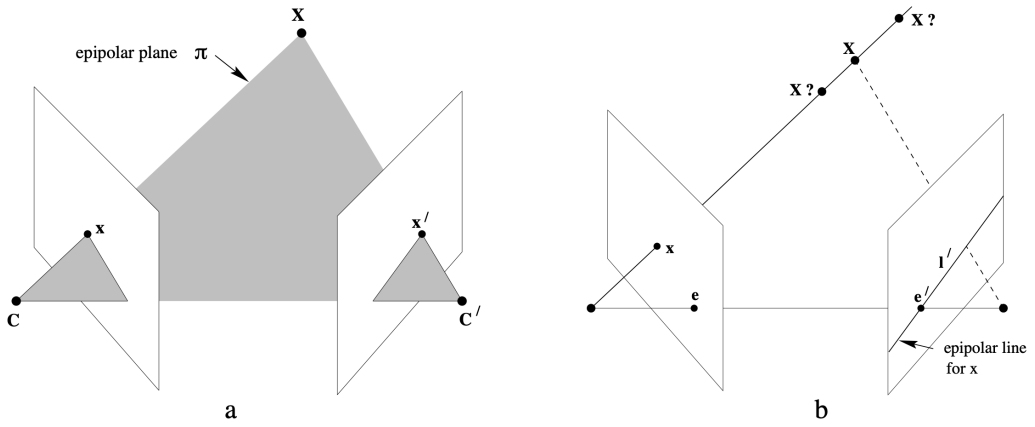

Figure 3.7: Point correspondence geometry.

- 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(3.7) 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.

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

Let's consider the following scenario,reported in Figure(3.8): 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 $\mathbf{t}$.

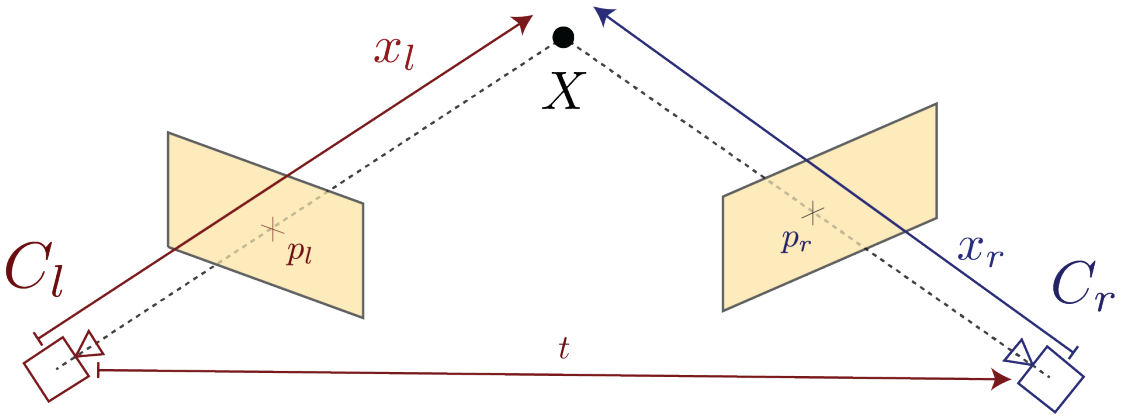

Figure 3.8

We can extract the Essential Matrix by making the following considerations:

$$x_l \cdot (t \times x_l) = 0 \tag{3.1}$$

but x_l can be written as:

$$x_l = Rx_r + t \tag{3.2}$$

So Equation(3.1) becomes:

$$x_l \cdot (t \times (Rx_r + t)) = x_l \cdot (t \times Rx_r) \overset{a}{=} x_l^T \, [t]_\times \, Rx_r = 0 \tag{3.3}$$

where equality (a) is due to the cross-product matrix notation[1]. We call Essential

[1]

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

Matrix the term $[t]_\times R$, such that:

$$x_l^T [t]_\times R x_r = x_l^T E x_r = 0 \tag{3.4}$$

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:

$$p_l = \frac{1}{z_l} K_l x_l \quad p_r = \frac{1}{z_r} K_r x_r \tag{3.5}$$

where z_i are the focal distances.We can substitute in Eq.(3.4) and obtain:

$$p_l^T z_l K_l^{-1^T} E K_r^{-1} z_r p_r = 0 \quad z_l, z_r \neq 0 \tag{3.6}$$

Since z_i are constants can be simplified, obtaining:

$$p_l^T K_l^{-1^T} E K_r^{-1} p_r = p_l^T F p_r = 0 \quad z_l, z_r \neq 0 \tag{3.7}$$

where F is the Fundamental Matrix. We can thus write the relation between the two matrices:

$$E = K_l^T F K_r \tag{3.8}$$

3.1.3 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[2]using singular value decomposition.

3.1.4 Multi-view Structure from Motion

Now that we have grasped the basics to find images correspondances, let's see how we can relate multiple image to recover the structure of a scene. The last stage is called *bundel adjustment* and it is a iterative algorithm used to adjust structure and motion parameters by minimising a cost function. The possible methods for bundle adjustment are:

---

[2]

$$[t]_\times = U W \Sigma U^T \tag{3.9}$$

$$R = U W^T V^T \tag{3.10}$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

- **Sequential**: which work by considering an additional images at each time, extending in this way the initial reconstruction.

- **Factorization**: work 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. Minimising an appropriate cost function we can obtain a clean model as in a Linear Regression.

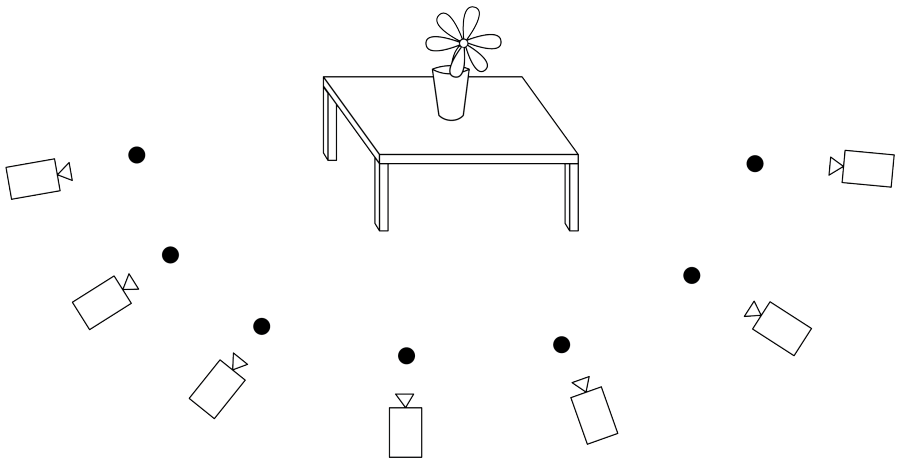

Figure 3.9: Multiple View Scenario.

3.1.5 COLMAP

Figure 3.10: Building Rome in one day

Abandoning theory and moving on to practice, let's see COLMAP.

"COLMAP is a general-purpose, end-to-end image-based 3D reconstruction pipeline (i.e., Structure-from-Motion (SfM) and Multi-View Stereo (MVS)[3]) with a graphical and command-line interface. It offers a wide range of features for reconstruction of ordered and unordered image collections. The software runs under Windows, Linux and Mac on regular desktop computers or compute servers/clusters. COLMAP is licensed under the BSD License"[23].

The concepts we explained in the theoretical part are not always applicable In real world, or their results are not quite satisfying. In literature a vastity of authors tried and succeded in obtaining refined algorithms for specific scenarios. Anyway 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 from the authors in [24, 25].

Usage

The usage of COLMAP is pretty straight forward. 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 (3.1).

Listing 3.1: 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

```

Anyway the program offers ample freedom to the single usage of the various steps need for *SfM* or *MVS*. Here I report the actual pipeline that I used for the extraction of the pointclouds for our experiments.

Listing 3.2: COLMAP Single Commands

```
    #!/bin/bash
```

[3] *Even if they seem similar, these two pipelines have different goals. In fact, SfM The primary goal of SfM is to estimate the 3D structure of a scene and camera poses (positions and orientations) simultaneously from a set of 2D images taken from different viewpoints. MVS,on the other hand, is specifically focused on dense 3D reconstruction. It involves estimating the depth or 3D coordinates for every pixel in the images to create a detailed 3D model with meshes and textures.*

```
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
```

```
43
44
```

3.1.6 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*. As we have previously seen, the projection of a scene on the bidimensional image plane of a camera inevitably lose the three-dimensional structure of the scene. In this way we can no more use optics laws, since we do not have enough informations 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 good from neural networks, in particular convolutional or transformer based networks (See Section (2)), trained on large amount of data, which are nowadays growing in number and quality.

In particular the advent of transformers has marked a new state of the art, not only in natural language processing, but also in dense prediction. As shown and stated in [26], 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 yieds 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 motivation is due to the different functioning of the two networks. Infact, both methods implies an encoder-decoder structure but the nature of convolutional layer limits its performance. As explained in Sec.**??** convolutional networks progressively downsample the input image to extract features at multiple scales. Unfortunately in dense prediction tasks, keeping feature resolution and granularity showed to be essential, since the decoder can't recover the initial informations from the compressed ones. "*Unlike fully-convolutionals networks, the vision transformer backbone foregoes explicit downsampling operations after an initial image embedding has been computed and mantains a representation with constant dimensionality throughout all processing stages*"[26].

The second motivation that made us choose for the transformer-based architecture is its versatibility and generalizability. In [27] the authors introduce a method that enable 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. In particular they experimented with

eleven datsets which consists of RGB images with corresponding depth annotation of some form:

- Training: DIML Indoor,MegaDepth,ReDWeb,WSVD, 3D Movies
- Testing:DIW,ETH3D,Sintel,KITTI,NYUDv2,TUM-RGBD

Each single dataset has its own characteristic and has its own biases and problems. Mixing them during the train phase allows to mitigate those problems and obtain a good generalization. To evaluate the generalization they used the *Zero-Shot Cross-dataset Transfer* which consists in evaluating on datasets that were not seen during training. This proved to be extremely succesful.

Dataset	Indoor	Outdoor	Dynamic	Video	Dense	Accuracy	Diversity	Annotation	Depth	# Images
DIML Indoor	✓			✓	✓	Medium	Medium	RGB-D	**Metric**	220K
MegaDepth		✓	(✓)		(✓)	Medium	Medium	SfM	No scale	130K
ReDWeb	✓	✓	✓		✓	Medium	**High**	Stereo	No scale & shift	3600
WSVD	✓	✓	✓	✓	✓	Medium	**High**	Stereo	No scale & shift	1.5M
3D Movies	✓	✓	✓	✓	✓	Medium	**High**	Stereo	No scale & shift	75K
DIW	✓	✓	✓			Low	**High**	User clicks	Ordinal pair	496K
ETH3D	✓	✓			✓	**High**	Low	Laser	**Metric**	454
Sintel	✓	✓	✓	✓	✓	**High**	Medium	Synthetic	(Metric)	1064
KITTI		✓	(✓)	✓	(✓)	Medium	Low	Laser/Stereo	**Metric**	93K
NYUDv2	✓		(✓)	✓	✓	Medium	Low	RGB-D	**Metric**	407K
TUM-RGBD	✓		(✓)	✓	✓	Medium	Low	RGB-D	**Metric**	80K

Figure 3.11: Monocular datasets

Scene	Feature Extractor	Exhaustive Matcher	Mapper	Image Undistorter	Patch Match Stereo	Stereo Fusion	Total Time
A							
B							
C							
D							
E							
F							
G							

Table 3.1: Your Table Caption Here

Figure 3.12: Top: input images. Middle: Inverse depth maps predicted by the presented approach. Bottom: corresponding point clouds rendered from a novel view-point.(Taken from [27])

A Head	A Second Head	A Third Head
Some text	Some really longer text	Text text text

Part II

Da Sistemare

3.2 Metrics

CHIEDERE COME MOTIVARE LA SCELTA DELLE NOSTRE METRICHE
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 [28]. Namely these are:

- **PSNR**
- **AP**

3.2.1 PSNR:Peak signal-to-noise ratio

The Peak Signal-to-Noise Ratio (PSNR) 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.

3.2.2 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. It measures the *precision-recall* trade-off of a model.

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

$$Precision = \frac{TP}{TP + FP} \tag{3.12}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.13}$$

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 3.13. 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.

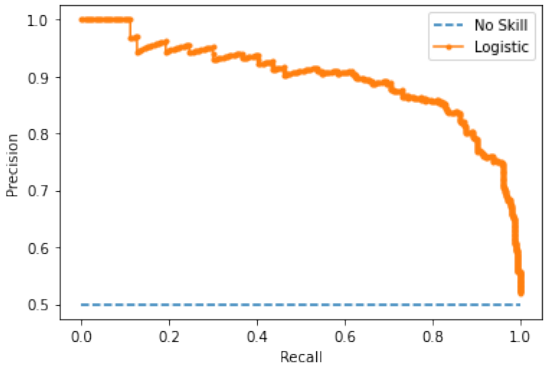

Figure 3.13: Example of Precision-recall curve.We can see how the bottom line model represents the worst a model can perform, e.g. predict every sample as it is coming from the same class,if the dataset is balanced. A better model would *tend* to the upper-right corner, which instead represents the best possible model, a model that have maximum precision and recall.

Scene	Feature Extractor	Exhaustive Matcher	Mapper	Image Undistorter	Patch Match Stereo	Stereo Fusion	Total Time
A							
B							
C							
D							
E							
F							
G							

Table 3.2: Your Table Caption Here

A Head	A Second Head	A Third Head
Some text	Some really longer text	Text text text

Appendix A

Galileo

```
    import os
    os.system("echo 1")
```

$\mathcal{O}\left(n \log n\right)$
`numpy`

Appendix B

Math Notation

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

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

Bibliography

[1] Dima Damen et al. «Scaling Egocentric Vision: The EPIC-KITCHENS Dataset». In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 3).

[2] 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. 3).

[3] 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. 3).

[4] 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. 3).

[5] 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. 3).

[6] 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. 3).

[7] 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. 3).

[8] 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. 3).

[9] 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. 3).

[10] 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. 3).

[11] 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. 3).

[12] 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. 3).

[13] 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. 3).

[14] 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. 3).

[15] 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. 3).

[16] Wolf and Dewitt. 2000; McGlone, 2004 (cit. on p. 7).

[17] Mathworks 2023. URL: https://www.mathworks.com/help/vision/ug/structure-from-motion.html#:~:text=Structure%20from%20motion%20(SfM)%20is,localization%20and%20mapping%20(vSLAM). (cit. on p. 8).

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

[19] 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).

[20] 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).

[21] 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. 11).

[22] 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).

[23] Colmap 2024. URL: https://colmap.github.io/index.html (cit. on p. 17).

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

[25] 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).

[26] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. «Vision Transformers for Dense Prediction». In: *ICCV* (2021) (cit. on p. 19).

[27] 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. 19, 21).

[28] 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 p. 23).