Université Saint-Joseph de Beyrouth
Faculté d'ingénierie
Faculté des sciences

Academic Year 2022-2023

Master II

Curriculum: Data Science

Final Thesis

Enhancing Temporal Consistency for Monocular Video Depth Estimation

By

Mohammad Falha

Project Director: **Mr. Issa Hammoud**

Academic Supervisor: **Mr. Youssef Bakouny**
Location: **7 Sensing Software, Paris, France**

Jury Members: **Mr. Maroun Chamoun, Mr. Toni Sayah, Mrs. Rima Kilany, Mrs. Gihane Mansour, Mrs. Katia Raya, Mr. Youssef Bakouny**

Date: **30-10-2023**

Abstract

Video depth estimation represents a crucial component in the field of computer vision, finding diverse applications in domains such as augmented reality, the creation of 3D reconstructions for autonomous vehicles, and various other domains. Nonetheless, the key challenge in formulating precise depth estimation solutions has been the issue of temporal inconsistency, particularly when these solutions are applied sequentially to individual frames of a video.

To address this challenge, numerous suggested methodologies have harnessed optical flow and dense motion estimation between consecutive frames. However, these approaches often necessitate substantial video training data and lead to computationally complex processes marked by latency issues that restrict their practical implementation. In light of this, the current project endeavors to introduce a robust metric for quantitatively assessing temporal consistency without relying on ground truth data. Additionally, the project implements cutting-edge techniques that conform to specific constraints.

Two distinct solutions were explored within this project: firstly, the incorporation of sparse Jacobian regularization into the network following the application of specific transformations on the image; secondly, the fine-tuning of the initial precise network to mimic the consistency of a consistent yet slightly inaccurate video.

Acknowledgments

I would like to extend my heartfelt gratitude to God for granting me strength, guidance, and blessings throughout this academic journey.

Special appreciation goes to my supervisor, Mr. Issa Hammoud, for his exceptional guidance, and valuable insights.

I would also like to thank my academic supervisor, Dr. Youssef Bakouny for his mentorship.

My deepest thanks to my family, whose love, encouragement, and sacrifices have been the cornerstone of my achievements.

Contents

List of Figures

List of tables

Symbols

TC	Temporal Consistency
$MAPE$	Mean Absolute Percentage Error
MSE	Mean Square Error
MAE	Mean Absolute Error
EMA	Exponential Moving Average
BS	Batch Size
LR	Learning Rate

Chapter 1: Introduction

For a while, video depth estimation has been a tough challenge. This chapter begins by defining the problem and mentioning its main use cases. It then discusses the limitations and constraints of the solution. Following that, the project's objectives are presented, with a clear scope and key definitions. Finally, the chapter concludes by providing an overview of the remaining chapters.

1.1 Background

The term "AI" has evolved into a broad field, containing numerous domains, each of which continues to advance with the publication of hundreds of research papers annually. One of these domains is computer vision, involving various tasks related to image processing, generation, and comprehension. These tasks include object detection, face recognition, object tracking, image classification, and medical image analysis.

The focus of this project is a specific application within computer vision: video depth estimation. This application primarily involves generating a new video from an existing one that measures the depth of each pixel with respect to the camera, with each object in the video assigned a unique color corresponding to its specific depth. This technique has multiple real-world applications, including its use in autonomous vehicles, since depth video estimation plays a critical role in object detection, distance measurement, and the comprehension of the 3D environment surrounding the vehicle. Another application belongs to the realm of Augmented Reality (AR), where depth videos enable the realistic placement of digital elements and virtual objects into real-world videos. Furthermore, it finds relevance in 3D modeling and medical image reconstruction.

This project builds upon an existing solution for image depth estimation and aims to adapt and extend this solution to apply it to videos.

1.2 Context

The problem addressed by the project is to improve the temporal consistency of depth estimation. While a solution exists for images and yields decent results, challenges arise when applying this solution to videos frame by frame. Consequently, the resulting video exhibits significant inconsistency issues, notably characterized by flickering. This inconsistency is mainly represented by irregular and significant variations in the same pixels in consecutive frames. Handling this problem involves many constraints and difficulties that must be considered in any potential approach, as summarized below:

1. The project focuses on Monocular depth estimation, which means that it works with only one camera to estimate depth for scenes, unlike other

approaches that use multiple cameras, such as stereo depth estimation with two cameras. The project aims to develop a model for depth estimation based on a single image.

2. The available training data consists of discrete indoor scenes rather than video or consecutive images. There are no consecutive frames to train the model.
3. The project should avoid making crucial modifications to the network, such as changing the architecture or the existing solution. Instead, it should aim for compatibility with the existing solution with limited modifications, such as adding suitable loss functions or augmentation techniques.
4. The project's final goal is to create a mobile-friendly application. This goal introduces constraints related to maximum execution time, memory footprint, and processor requirements. Any suggested solution must be lightweight to meet these constraints.
5. While addressing the temporal consistency problem, the project must also preserve a minimum level of accuracy, which may vary depending on the specific case. Different solutions may improve temporal consistency at the expense of other crucial metrics, such as MAPE and edge metrics. These metrics are essential for validating any solution due to their impact on the performance and accuracy of the model.

1.3 Purposes

This project aims to enhance temporal consistency in video depth estimation. Multiple deep learning approaches were explored, with introducing novel contributions through changes in the loss functions and fine-tuning (as detailed in Chapter 3). These approaches will be evaluated following extensive research to select a suitable metric and implement it to benchmark the performance of various models.

Throughout the development process, numerous model training iterations were performed with different configurations, including alterations to loss functions, training data size, number of epochs, model parameters, layers, and skip connections.

1.4 Significance, Scope, and Definitions

This project's contribution lies in modifying existing solutions to enforce temporal consistency constraints in a manner compatible with all the mentioned difficulties and constraints.

The proposed approach is designed to function with cameras equipped with depth sensors. It's important to note that the model is trained on indoor scenes data, which may limit its performance in outdoor scenes unless trained on a more extensive dataset that includes outdoor scenes.

The methodology employed in this project involves exploring the mechanics of the existing solution and understanding how the architecture functions. Subsequently, naive post-processing techniques were implemented to establish a baseline, followed by state-of-the-art methods that align with the constraints and improve temporal consistency. This research will include novel approaches and recent publications addressing similar challenges. The selected findings will then be implemented, and the results tested.

In the context of the project, the following are definitions for frequently used terms:

- Temporal Consistency: This term refers to the consistency in a model's predictions over time, ensuring that the video appears smooth without flickering or sudden jumps in pixel values.
- MAPE (Mean Absolute Percentage Error): MAPE stands for "Mean Absolute Percentage Error" and is calculated by determining the mean of the absolute percentage differences between the predictions and the target values. It is often expressed as “mean (|(estimated pixel – ground truth) / ground truth|)”. We use MAPE instead of other metrics, such as MAE (Mean Absolute Error), to provide a more accurate estimate of the model's output. Other metrics like MAE reflect the absolute difference between the ground truth pixel and the estimated one, which may not be ideal for our case, given the relatively large range of depth values, spanning from 0 to 12000 mm. In contrast, MAPE measures the percentage by which the estimated image deviates from the original one. For example, a 4% MAPE means that the model's output images differ from reality by 4%. For instance, having an MAE of 20000 mm for the output image may not provide a robust indicator, whereas a 4% MAPE serves as a significant metric for our work.
- Sparse Image: In the model, two images are used as input, the RGB image and the sparse depth image. The sparse depth image contains depth data for only a subset of pixels, while the remaining pixels have null values. This image is simulated from a DToF (direct time-of-flight) sensor, as further detailed in Chapter 3.
- Edge Metric: This is a quantitative measurement that assesses the dissimilarity in the edge regions between the ground truth and the model's prediction. Lower values indicate a better match with the ground truth, and higher values suggest the opposite.

- Optical Flow: Optical flow refers to the motion of pixels in consecutive frames. It calculates a vector field where each vector describes the motion or displacement of each pixel.

1.5 Thesis Outline

The report outline will follow this structure:

- Chapter 2: Literature Review

Discusses the papers that present state-of-the-art approaches for enhancing temporal consistency in video depth estimation. Evaluate their applicability to our case and explore their requirements and limitations.

- Chapter 3: Research Design:

This chapter presents the project's methodology and how it addresses the problem. It also explains the data used, mentioning its source, and finally, it discusses the project's limitations.

- Chapter 4: Implementation

  Discusses the resources used, both in terms of software and hardware, and presents the approaches, as well as the underlying logic and mathematics, and it includes four main topics:

- Explain the network architecture used in video depth estimation.
- Propose and Implement a Metric to Evaluate Temporal Consistency.
- Apply Two Different Post-Processing Approaches: These approaches aim to smoothen the video and enhance temporal consistency. And then test their effects on other metrics.
- Propose Two Deep Learning Approaches: These approaches are drawn from two different papers and implemented with some modifications.

- Chapter 5: Results

  Present the results of all the approaches and different configurations.

- Chapter 6: Analysis:

  Analyze the results of the models and provide a basis for selecting the best trade-off based on the use cases.

- Chapter 7: Conclusion

State the best results achieved in the project and outline potential enhancements and choices that can be applied to improve the results.

1.6 Action Plan

Table 1 displays the tasks accomplished during the 7-month internship at 7 Sensing Software Company, along with the duration allocated to each task and their respective dates.

Task	Duration	Date
Understand the existing solution for depth estimation and reproduce the results.	3 weeks	March 23rd - April 15th
Search/Propose a metric to quantify well the temporal inconsistency so we can estimate the superiority of different solutions	3 weeks	April 15th - May 10th
Try simple post-processing strategies for temporal consistency to get a baseline	3 weeks	May 10th - May 30th
Study and document state of the art solutions and choose one or two	4 weeks	May 30th - June 30th
Implement/compare the state-of-the-art methods that are mobile-friendly	16 weeks	June 30th - October 27th

Table 1: Task Overview and Time Allocation

Chapter 2: Literature Review

Numerous studies and research papers have discussed the retrieval of depth images from single images and proposed novel approaches to enhance temporal consistency without compromising the accuracy of the model. These approaches can be categorized into several groups. Some of them focused on data quality and utilized consecutive frames during model training to replicate consistency in the output.

Other approaches introduced entirely new deep networks designed to enforce consistency between frames. Some papers proposed the use of computationally efficient regularization terms that do not require significant modifications to existing solutions. In addition, certain papers explored hybrid approaches, which involved adding a new network responsible for enforcing consistency.

Furthermore, an essential part of the study is to select a robust metric and implement it to evaluate different approaches and their effectiveness in improving consistency. This evaluation should provide quantitative measures of their performance.

Given the numerous constraints associated with the solution we are seeking, an in-depth review of several state-of-the-art papers was conducted to identify a temporal consistency metric that can provide accurate quantitative measures and to identify compatible solutions suitable for our specific case.

2.1 Temporal Consistency Metric

In the context of evaluating temporal consistency, several metrics have been proposed in the literature. Zhang, Shen, Li, Cao, Liu, and Yan (2019) [4] introduced two metrics - temporal change consistency (TCC) and temporal motion consistency (TMC). These metrics are known for providing robust and realistic measurements, but they depend on consecutive ground truth depth images, which are unavailable in our situation.

Varghese, Bayzidi, Bär, Kapoor, Lahiri, Schneider, Schmidt, Schlicht, Hüger, and Fingscheidt (2020) [5] conducted a study that introduced an unsupervised temporal consistency metric designed for video segmentation. However, this metric is more specialized as it focuses on segmentation tasks, quantifying the similarity or overlap between segmentation masks in consecutive frames. This is different from our specific goal, which involves estimating continuous depth values (distance) for each pixel in an image within the 0-12000 mm range, where the model must predict real numbers (depth values) rather than discrete categories as in segmentation.

Siyuan Li, Luo, Zhu, Zhao, Li, and Shan (2021) [2] presented a new metric for testing temporal consistency - the Temporal Consistency (TC) metric. This metric seeks to replicate the variations between depth maps and the original images using optical flow operators. Notably, the advantage of this metric is that it doesn't rely on ground truth data, making it a suitable choice for our specific case. A more detailed explanation of this metric will be provided in the following chapter, highlighting its relevance to our research objectives.

2.2 Enforcing Temporal Consistency in Video Depth Estimation

In their research on real-time video depth estimation, Zhang, Shen, Li, Cao, Liu, and Yan (2019) [4] tackled the problem of temporal inconsistency by proposing a spatial-temporal CSLTM (ST-CLSTM) structure. This approach involves a novel architecture comprising three main components. The first component is a spatial feature extraction network, which includes an encoder, decoder, and a multi-scale feature fusion module (MFF). This network takes consecutive frames as input and aims to extract features from the images. The second component is the CSLTM network, which leverages temporal correlations between these features. Finally, the last component utilizes two primary loss functions - spatial and temporal loss - to minimize the difference between the ground truth depth and the estimated depth while ensuring temporal consistency. However, this solution may not apply to our case due to its requirement for significant modifications to the existing solution and architecture, which is a constraint. Additionally, it necessitates video training data, which is unavailable.

In their study on depth estimation in video sequences, Siyuan Li, Luo, Zhu, Zhao, Li, and Shan (2021) [2] introduced a novel approach to enhance temporal consistency in video depth estimation. Along with proposing a compatible TC metric, the paper suggested incorporating a TC loss function into the network. This method uses optical flow between consecutive frames during training and results in significant improvements in temporal consistency. Nevertheless, it does require consecutive frames as training data, making it incompatible with our specific case.

Eilertsen, Mantiuk, and Unger (2019) [3] proposed a single-frame regularization method to improve the temporal stability of CNNs. This approach involves adding regularization to the loss function to enforce temporal consistency. The key contribution of this approach lies in its ability to provide a solution that doesn't necessitate video training data or dense motion estimation between frames, making it computationally efficient. The paper introduced three regulations to ensure that small variations in input lead to small variations in the output: stability, transform invariance regularization and sparse Jacobian regularization. The results of their work, which were applied to various tasks such as colorization and HDR reconstructions, demonstrated its generality and applicability beyond specific tasks.

Lei, Xing, and Chen (2020) [6] proposed a solution for blind video temporal consistency using a deep video prior, this method begins with two videos: one that is the original and the other that has been processed using a particular image processing algorithm, like a depth estimation algorithm. The initial step involves training a convolutional neural network (CNN) to mimic the behavior of the image processing algorithm. To ensure accuracy, this training procedure focuses on decreasing the reconstruction error between the processed video and CNN’s output. A temporal consistency loss is added to the training objective to improve the results by instructing CNN to generate temporal consistent results. After CNN has been trained successfully, it can be used to produce a processed video that is temporally consistent. The method's independence from labeled data stands out as one of its primary advantages, rendering it applicable to a variety of tasks, as demonstrated by the authors during their evaluation of the technique across seven computer vision tasks: dehazing, denoising, deshadowing, deraining, super-resolution, style transfer, and colorization. Their findings indicate that their method outperforms state-of-the-art techniques quantitatively and perceptually.

2.3 Summary and Implications

As discussed above, the state-of-the-art approaches I studied encompass various methods for addressing the problem, each with unique constraints and limitations. Therefore, I excluded papers that required video training data or a complete change in the architecture. Instead, I selected papers that adhered to the project's constraints [3][6], and valuable contributions were added by this project, such as modifying the regularization loss function and fine-tuning the network. Additionally, Lei et al.'s suggestion for the TC metric [2] was adapted and implemented. These enhancements will be discussed in detail in the forthcoming chapters.

Chapter 3: Research Design

3.1 Methodology and Research Design

3.1.1 Methodology

This section outlines the methodology employed in the study to improve image depth estimation and enhance temporal consistency in the results. The methodology is divided into several key steps as listed below:

A- Understanding the Existing Solution: Begin by comprehensively analyzing the existing image depth estimation solution. Examine the encoder-decoder architecture employed, focusing on the architecture size, types of layers used, and the loss functions in place.

B- Proposing and Implementing Temporal Consistency Metric: Develop a novel metric to quantify the temporal consistency of the depth estimation results. Implement this metric and rigorously test its effectiveness.

C- Baseline Temporal Consistency Enhancement: Apply simple post-processing strategies to establish a baseline for improving temporal consistency. Investigate the impact of Mean kernel and exponential smoothing techniques with varying levels of severity. Assess the behavior of other metrics in conjunction with these post-processing strategies.

D- Studying State-of-the-Art Methods: Explore approaches to address the temporal consistency issue in depth estimation. Investigate and test two potential solutions:

   a. Impose sparse Jacobian regularization after applying transformations to the training frames, enabling treated frames to be treated as subsequent frames.

   b. Fine-tune the network to mimic a consistent depth video while maintaining high accuracy.

E- Model Configuration and Training: Execute multiple models runs with different configurations, varying parameters such as the weight of regularization, batch size, and data augmentation. This step aims to study the impact of these configurations on the results and the temporal consistency metric.

F- Comparative Analysis and Selection: Compare the results obtained from different model runs. Evaluate the trade-offs between different configurations and methodologies. Select the best-performing model based on the established trade-offs and the desired level of temporal consistency.

By following this comprehensive methodology, the study aims to improve image depth estimation while addressing temporal consistency issues, while preserving robust and accurate results.

3.1.2 Research Design

In this study, the research approach is quantitative, with the primary aim of improving the temporal consistency metric. The main independent variables encompass parameters such as the weight of regularization, batch size, and augmentation, while the main dependent variables include the Temporal Consistency (TC) metric, Mean Absolute Percentage Error (MAPE), and edge metrics. The hypothesis adopted by this project proposes that post-processing approaches will enhance TC but may lead to a decline in accuracy metrics. In contrast, the imposition of regularization that enforces temporal consistency is expected to contribute to the highest TC, without causing significant effects on other metrics.

3.2 Instruments

The instruments utilized in this study featured advanced encoder-decoder architecture, and a U-net architecture was employed in the second approach, incorporating numerous layers and metrics. Key metrics considered in the evaluation process include L1 or MAE (Mean Absolute Error) L2 or MSE (Mean Square Error) MAPE (Mean Absolute Percentage Error) Edge metric These metrics were used to assess the performance and effectiveness of the models and post-processing strategies in achieving the research objectives.

3.3 Data

The data used in this study is sourced from the open-access dataset "SceneNet RGB-D”, as published by Roberts, Ramapuram, Ranjan, Kumar, Bautista, Paczan, Webb, and Susskind (2020) [7]. This dataset comprises synthetic data consisting of

77,400 images generated from 461 indoor scenes. Each image is accompanied by detailed per-pixel labels and corresponding ground truth geometry. Notably, these images are created in a manner that adheres to physical constraints, closely resembling real-world data.

3.4 Ethics and Limitations

This research adheres to ethical guidelines. Ethical approvals were obtained from the company to use their generated data and share the report along with its details.

It is important to acknowledge that the research may have limitations, including potential errors in outdoor scenes or scenes with different lighting conditions that deviate from those present in the training dataset. Furthermore, it should be noted that the architecture is optimized for a specific type of depth sensor and may not be universally applicable to all sensor types, and the sensor used is designed for depth values that range from 0 to 7000 mm.

.

Chapter 4: Implementation

4.1 Hardware and Software Resources

During the phases of the project, several software and hardware tools were utilized, they are listed below with details:

- 2 GPUs, each 24 GB, were utilized for training the models. Some models were trained on a single GPU, while others, particularly those with large batch sizes, required both GPUs for training.
- MLflow was used to track the models and archive their configuration, results, and learning curves.
- Python was the programming language used, with VS Code used as the IDE.
- Jupyter Notebook was utilized for visualizing images and results, as well as for calculating metrics for specific outputs.
- TensorFlow framework was employed for designing the network.
- OpenCV framework (cv2) was used to implement optical flow in the tc metric.
- The project was hosted on Linux as OS on the cloud.
- Git was used on Bitbucket as a version control tool.

4.2 Architecture Overview

The fundamental concept of the encoder-decoder architecture implemented in the depth estimation solution revolves around its dual components: the encoder and the decoder (a visual representation of the architecture is shown in Figure 1). Within this framework, the encoder takes in an RGB image and a sparse depth image as its input, adeptly extracting high-level features. These extracted features are then passed to the decoder, which utilizes them to generate a detailed depth map. The model undergoes rigorous training on a comprehensive dataset comprising RGB and Spare images and corresponding ground truth depth maps, intending to minimize the error between predicted and actual depth maps. Once the model completes its training phase, it is ready to accurately estimate the depth of previously unseen images.

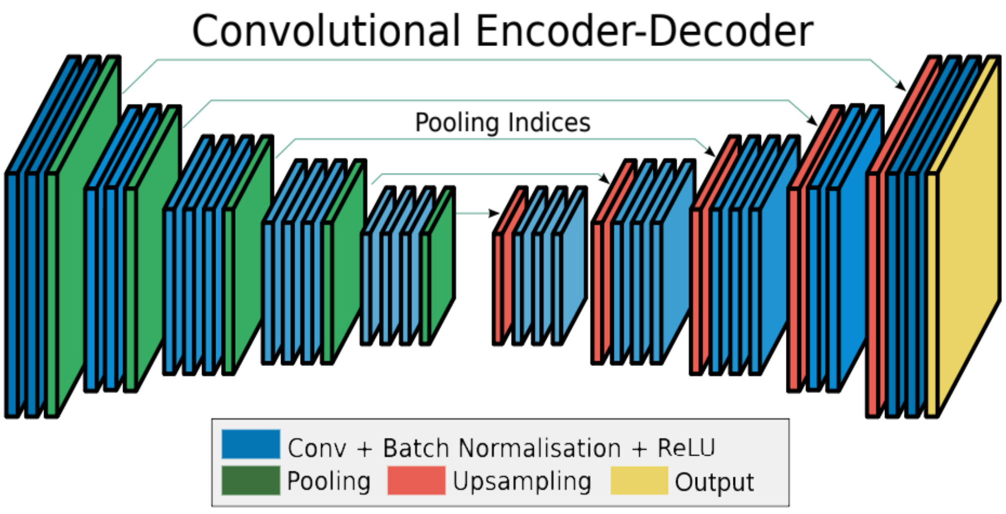

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Output

Figure 1 A generic encoder-decoder architecture commonly employed in computer vision tasks. Source: Badrinarayanan, Kendall, and Cipolla (2016) [1]

The model -as mentioned- has two distinct inputs: the first input is an RGB image with (256, 192, 3) dimensions, while the second input is a sparse depth map image, emulating data from the sensor, and it has dimensions of (256, 192), Following a standard preprocessing pipeline that includes clipping and normalization, each image proceeds through a Convolutional layer. To enhance the model's capacity for information fusion, an ADD layer is employed after the aforementioned layers. The ADD layer conducts element-wise addition, thereby combining the pixel values from the RGB image with their corresponding counterparts in the sparse image. Following each Convolutional layer, a batch normalization layer is incorporated, which not only accelerates the learning process but also serves as a regularization technique to avoid overfitting.

In conjunction with the batch normalization layer, an activation function, specifically Reu is applied. Notably, certain convolutional layers are divided into two pathways, which are later merged. This design, known as skip connections, promotes feature extraction in parallel, heightening prediction accuracy. Furthermore, skip connections address the vanishing gradient problem and facilitate the transfer of information across distant layers, preserving critical features. Figure 2 displays an example of two images, along with their corresponding inputs, ground truth, and predictions.

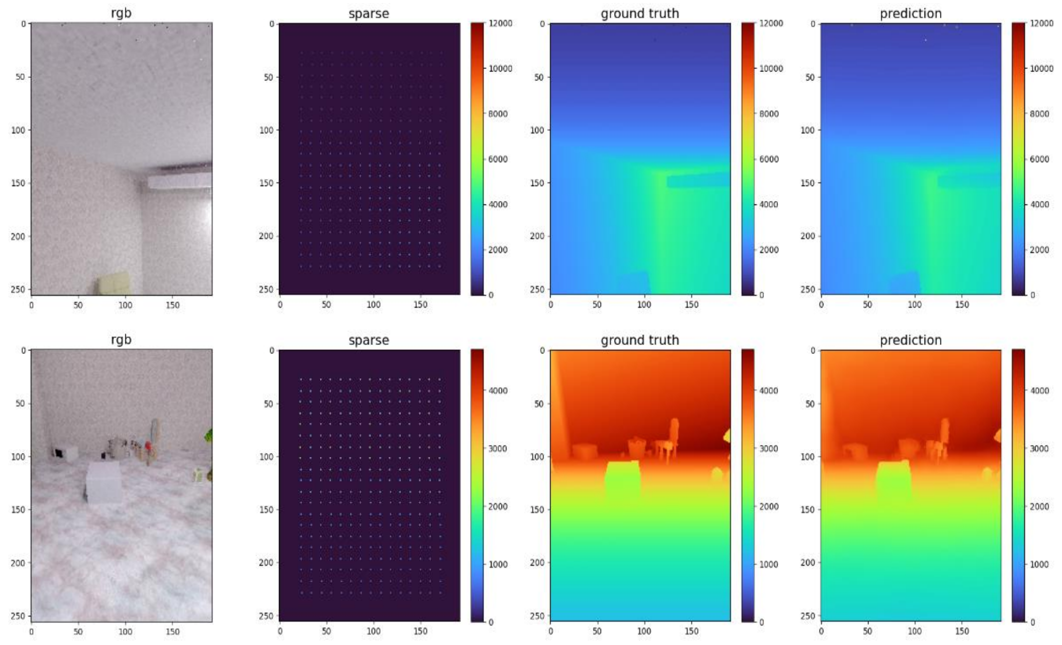


Figure 2: Example of 2 images where Input Images, Ground Truth, and Predictions are displayed respectively.

4.3 Temporal Consistency Metric

The first stage in addressing the problem is to propose a metric for measuring temporal consistency in videos. This metric will enable the examination and evaluation of different solutions to determine their impact on temporal consistency.

During this step, research papers suggesting TC metrics were carefully reviewed [2][4][5], in summary, the main approach underlying the metrics suggested was the same, as they all employed optical flow, a technique that describes the motion variation between consecutive images and returns an operator that describes each pixel motion with a vector of two components.

The fundamental concept behind the TC metric's mechanism as shown in Figure 1 involves calculating the optical flow between the original two consecutive RGB frames, therefore, this calculated operator is applied to the depth image of the first RGB frame, and the output is then compared to the second depth image. The ideal case is a zero-difference, indicating that the motion variation between two consecutive depth maps is equal to that of their corresponding RGB images. Consequently, a smaller difference between two depth images indicates a more consistent video and vice versa.

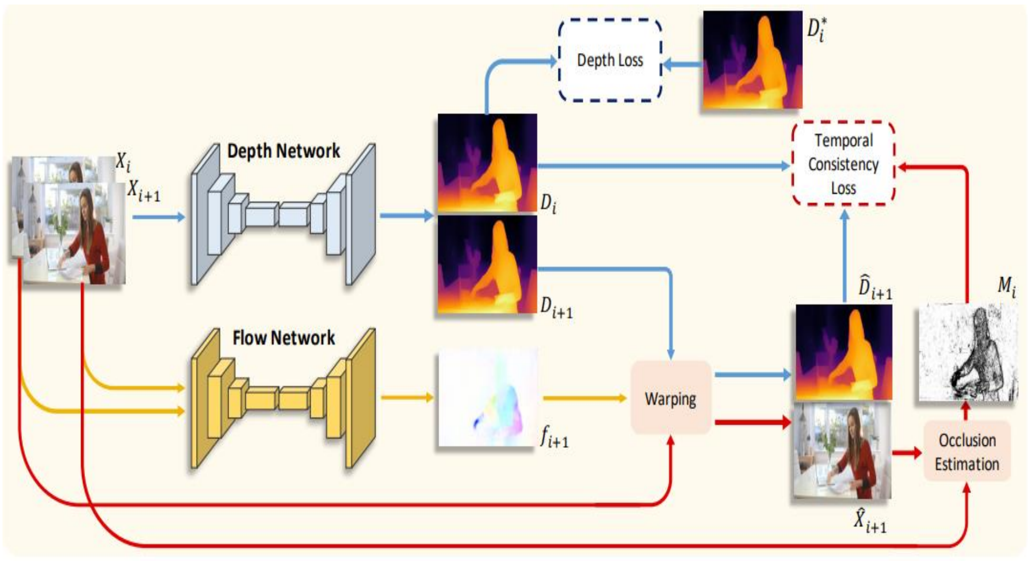


Figure 3 the mechanism of the implemented TC metric. Source: Li, S., Luo, Y., Zhu, Y., Zhao, X., Li, Y. and Shan, Y. (2021) [2]

In this project, the metric suggested by Li, S., Luo, Y., Zhu, Y., Zhao, X., Li, Y., and Shan, Y. [2] was used. Equation 1 states how the TC between two frames is calculated, and Equation 2 states that the overall TC corresponding to the whole video is calculated by adding the TCs of all frames and dividing it by the number of frames. Where the variables' definition of the used equation is as follows:

M_i: stands for the non-occlusion mask (it is also called valid mask or visible mask), its role is to exclude the occluded regions from the analysis, which ensures that the temporal consistency metric is only calculated on valid pixels.

D_i: Estimated depth map for the X_i frame.

$\widehat{D}_{i+1}$: output of the wrap operator after applying the optical flow tensor on the D_{i+1} frame.

The last part of the equation represents a condition, which states that if the difference between D_i and $\widehat{D}_{i+1}$ is smaller than the value of threshold it returns 1 else it returns zero while the value of the threshold varies from one research paper to another with examples such as 1.2,1.25,1.25^2 and 1.25^3 … [8].

$$the\ rTC_i = \frac{1}{\sum(M_i == 1)} M_i \cdot \left[Max\left(\frac{D_i}{\widehat{D}_i}, \frac{\widehat{D}_{i+1}}{D_i} \right) < threshold \right] \qquad \text{Equation 1}$$

$$TC = \frac{1}{n}\sum_{i=1}^{n} rTC_i \qquad \text{Equation 2}$$

The output of equation 1 represents the ratio of pixels that fall within an acceptable range determined by the threshold. For instance, if the value of the TC metric equals 0.9, this means that 90% of the predicted pixels have a modest (acceptable) variation. In summary, the greater the metric value in Equation 2, the greater the level of temporal consistency.

4.4 Post-Processing Approaches

To address the inconsistency problem and test various solutions for their validity, as well as to establish a starting point, post-processing approaches were developed. These approaches serve as baselines to examine the extent to which temporal consistency can be improved and the resulting impact on other metrics. This analysis aims to determine the optimal trade-off that can be achieved and whether it is acceptable or not. To achieve this, two techniques were employed: the Mean Kernel method and exponential smoothing. These techniques result in a smoothed video with reduced differences between consecutive frames, thereby leading to a more consistent video stream.

1- Mean Kernel Method: The underlying concept of this smoothing method involves adding a batch of images -determined by the kernel size (typically 2, 3, or 4)- elementwise. The resulting summation is then divided by the kernel size, and the obtained value replaces the corresponding pixel value as explained in Equation 3.

$$smoothed\ image = \frac{1}{kernel}\sum_{k=0}^{kernel-1} images[i-k] \qquad \text{Equation 3}$$

2- Exponential Smoothing Method: This post-processing technique applies smoothing to the video frames, causing each pixel's value to depend on its original value as well as the values of the same pixel in previous frames. In equation 4, the parameters "1-α" and "α" determine the respective contributions of the current image and earlier images to the resulting image.

$$EMA_t = (1 - \alpha) \times image_t + \alpha \times EMA_{t-1} \qquad \text{Equation 4}$$

4.5 Deep Learning Approaches

After investigating post-processing approaches to serve as a baseline, 2 deep learning solutions were investigated and implemented, depending on deep research in the state-of-the-art methods.

4.5.1 Sparse Jacobian Regularization

This solution was introduced in [3], which is a lightweight approach that adheres to all the constraints explicitly mentioned in the introduction. The solution addresses the challenge of not having consecutive frames as training data to enforce consistency between them through a loss function. Instead, we introduce a transformation for each frame and treat the transformed frame as if it were the second frame. This transformation includes translation, rotation, zooming, and shearing to simulate the usual variations between consecutive frames in the real world.

Equation 5 represents the loss function of the regularization used. Here, Δx represents the effect of one of the transformations on the input image, y(x) is the ground truth from the training set corresponding to x, and T(x) = x + Δx, T(y) = y (x + Δx), and the function f represents the depth estimation algorithm. The equation shows that the loss is minimized when the prediction error for the transformed image and its depth output is similar to the prediction error for the original image and its depth output.

$$L\,Jacobian = \left\|\left(\mathrm{f}\left(\mathrm{T}(\mathrm{x})\right) - \mathrm{f}(\mathrm{x})\right) - \left(\mathrm{T}(\mathrm{y}) - \mathrm{y}\right)\right\|_2 = \left\|\left(\mathrm{f}\left(\mathrm{T}(\mathrm{x})\right) - \mathrm{T}(\mathrm{y})\right) - \left(\mathrm{f}(\mathrm{x}) - \mathrm{y}\right)\right\|_2 \qquad \text{Equation 5}$$

As demonstrated in the equation, the loss function between the transformed and original errors is typically represented as the mean square error (MSE). However, in this project, a modification was introduced, with the mean absolute error (MAE) loss being tested and yielding better results. This adjustment is attributed to the robustness of the MAE, which treats both small and large errors equally. In contrast, squared differences in the MSE give greater significance to larger errors [9]. Squared differences can lead to a bias in the results, favoring outliers with significant depth errors. This phenomenon explains why the MAE outperforms the MSE loss in terms of TC and other metrics, as we will demonstrate in the next chapter.

4.5.2 Fine-Tuning the Network to Preserve Consistency

The paper [6] is not entirely applicable in our case because we are aiming for a real-time solution without ground truth for the video. To address this, we introduced a novel modification for the paper's architecture that is shown at the right of Figure 4.

Initially, we designed a small U-Net network with approximately 400,000 parameters, featuring 2 inputs and one depth map output, the same as the structure of the original network. This small network was trained on 500 frames from the test video.

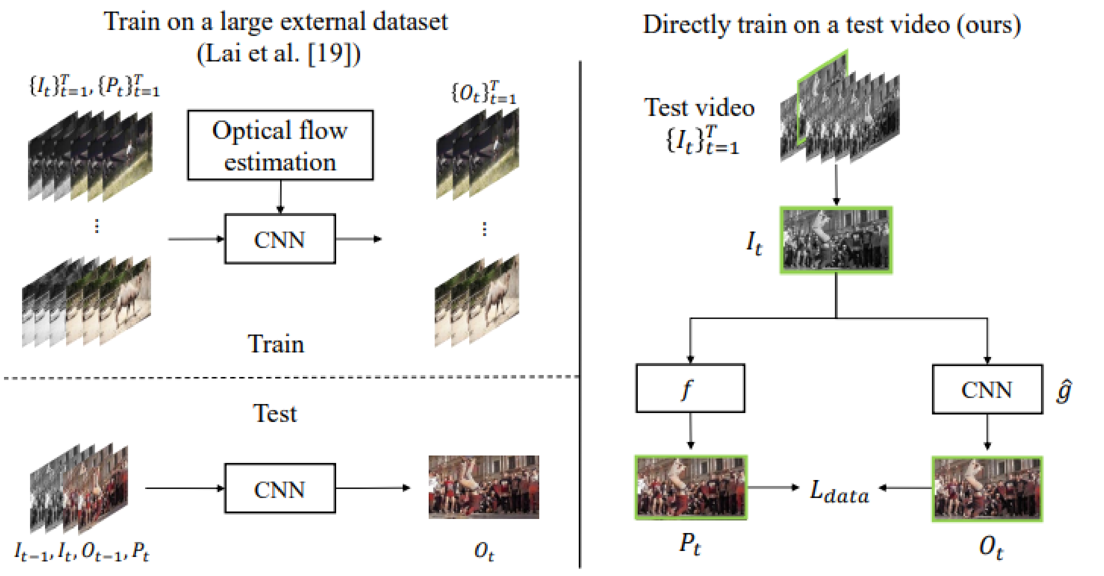

Train on a large external dataset
(Lai et al. [19])
$\{I_t\}_{t=1}^T, \{P_t\}_{t=1}^T$
Optical flow estimation
CNN
$\{O_t\}_{t=1}^T$
Train
Test
CNN
$I_{t-1}, I_t, O_{t-1}, P_t$
O_t
Directly train on a test video (ours)
Test video $\{I_t\}_{t=1}^T$
I_t
f
CNN
$\hat{g}$
L_{data}
P_t
O_t

Figure 4 The solution suggested by Lei et al. [6]

The assumption was that this small network would preserve temporal consistency and minimize flickering due to its simplicity, as confirmed by tests that demonstrated its ability to maintain consistency (approximately 91.5%). However, it exhibited poor mean absolute percentage error (MAPE) and edge metrics, as detailed in the next chapter.

Our proposed approach involved using the weights of the model resulting from the sparse Jacobian approach to enhance the temporal consistency as a pre-trained model. We froze all layers except the last one and trained the model solely on the 500 video frames, considering the predictions of the U-Net network (which resulted in a consistent but not accurate model) as the ground truth. Subsequently, the model was trained with only two loss functions - the Jacobian and the mean squared error (MSE) for 10 epochs.

The results and variations of TC, MAPE, and edge metrics were evaluated at the 1st, 5th, and 10th epochs. It is presumed that as the number of epochs increases, the TC metric will improve, while the MAPE and edge metrics will deteriorate, as the result tables in Chapter 5 will show.

Chapter 5: Results

5.1 Results of the Temporal Consistency Metric Implementation

After conducting this research, the metric was implemented in Python from scratch [Appendix A], and then tested over different videos to ensure the results were rational. Subsequently, the depth video generated by the current solution was evaluated for temporal consistency using this metric, and the results indicated a consistency rate of 79.5%. In the second part of the project, the goal was to enhance this metric through Postprocessing and Deep learning approaches and compare how they affected the TC metric and other metrics such as the MAPE and the edge metrics.

5.2 Results of Post-Processing Approaches

The post-processing methods explained in section 4.4 are implemented in Python, with different kernel sizes for the mean kernel method and different alpha values for Exponential Smoothing. An example of the effect of these methods on two consecutive frames is shown in Figures 5 and 6, respectively.

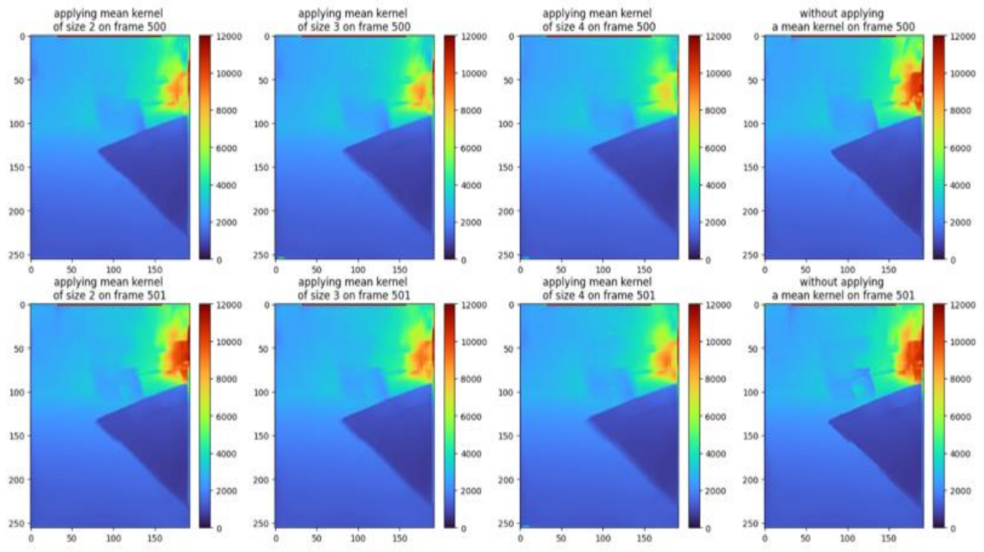


Figure 5 This image shows the estimated depth of two consecutive frames without and with different kernel sizes.

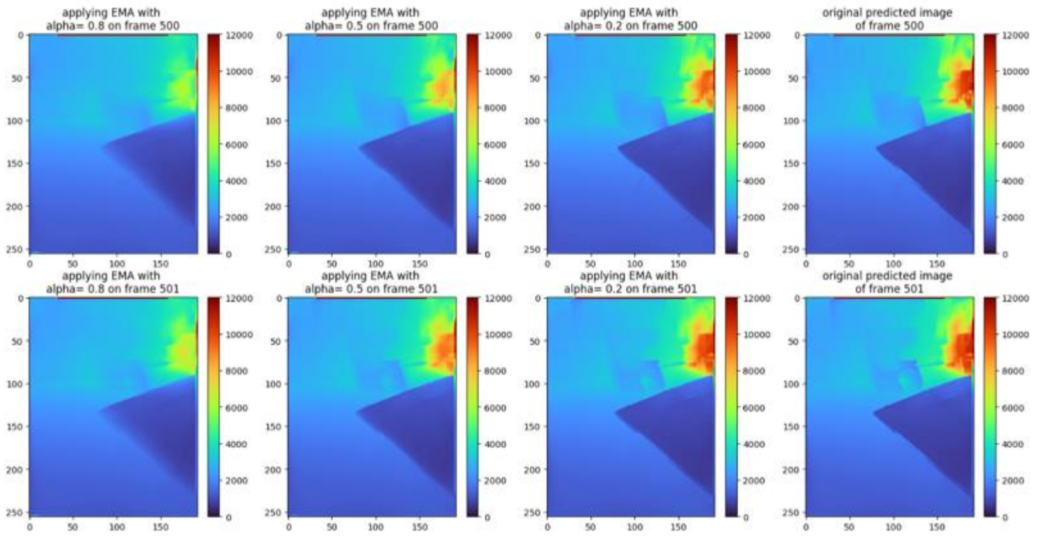


Figure 6 This image shows the estimated depth of two consecutive frames without and with different values of alpha.

The results of Applying the Mean kernel and Exponential Smoothing techniques are shown in Tables 2 and 3 respectively.

Kernel Size	**Temporal Consistency Mean**	**Temporal Consistency Median**	**MAPE 0-7 meters (gt is the original prediction)**	**Edge metrics (gt is the original prediction)**
No mean kernel is applied (the original output of the model)	**79.5%**	**82.3%**	**–**	**–**
Kernel size = 2	**82.4%**	**85.76%**	**2.97%**	**0.296**
Kernel size = 3	**83.1%**	**86.7%**	**4.24%**	**0.389**
Kernel size = 4	**83.4%**	**87%**	**5.27%**	**0.441**

Table 2 the results of applying the Mean Kernel method on TC, MAPE and edges metric.

Alpha	Temporal Consistency Mean	Temporal Consistency Median	MAPE 0-12 meters (gt is the original prediction)	Edge metrics (gt is the original prediction)
Alpha = 0 (the original output of the model)	79.5%	82.3%	_	_
Alpha = 0.99	95.7%	97.42%	46.95%	0.687
Alpha = 0.8	85.25%	89.9%	8.12%	0.465
Alpha = 0.5	83%	86.9%	3.32%	0.280
Alpha = 0.2	81.2%	84.35%	1.17%	0.112

Table 3 the results of applying the Exponential Smoothing method on TC, MAPE and edges metric.

5.3 Results of Deep Learning Approaches

5.3.1 Result of Imposing Spare Jacobian Regularization

The transformation suggested by Eilertsen et al. [3] was applied to the training images. A comparison was conducted among three models with a batch size of 64 and 100 epochs. The first model used no regularization, the second model utilized MSE as the loss function for regularization, and the third model used MAE as the loss function for regularization. The results are presented in Table 4 below.

Model Name	TC	MAPE	Edge Metric
TC015 without regularization	82.46%	1.95%	0.2582
TC016 Jacobian_mse	82.66%	2.37%	0.2638
TC017 Jacobian_mae	83.22%	1.72%	0.2538

Table 4 Comparison of models with/without Jacobian regularization

Afterward, to ensure the validity of the combinations we employed, several models were trained under various conditions, including those with or without augmentation, different batch sizes, and with or without regularization. The results are presented in Table 5.

Model name	Batch size	Augmentation	Regularization (MAE)	TC	MAPE	Edges error
TC 20	64	False	False	79.77%	1.71%	0.2396
TC 19	64	False	True	81.24%	1.63%	0.2442
TC 17	64	True	True	83.22%	1.72%	0.2538
TC 15	64	True	False	82.46%	1.95%	0.2582
TC 12	32	True	True	83.46%	2.26%	0.2515
TC 4	32	False	False	79.5%	1.72%	0.2430

Table 5 This table displays the Temporal Consistency, MAPE, and edge error for various combinations of loss functions, batch sizes, augmentation, and the application of Jacobian regularization.

To prevent overfitting, learning curves were monitored, ensuring that the training loss did not decrease while the validation loss increased. The curves for the best model (TC 17) are depicted in Figure 7.

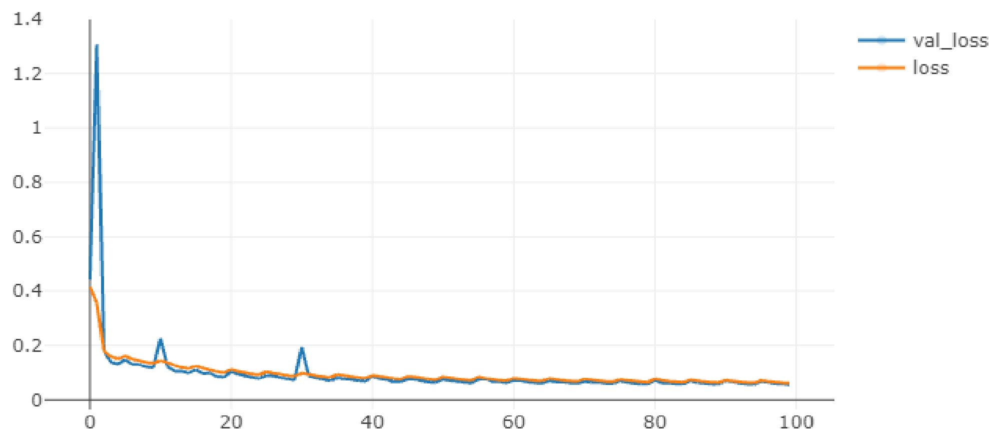


Figure 7 illustrates the variation in training and validation loss over 100 epochs in the TC 17 model.

5.3.2 Results of the Fine-Tuning Approach

Table 6 presents the results of the fine-tuning approach. In this approach, the consistent predictions of the U-net architecture were treated as the ground truth for the TC17 model. All of its layers were frozen, and only the last layers were kept for updates. The approach was tested under various configurations, including two learning rates and using weights from different epochs, as well as different batch sizes.

Model name	**Batch size**	**Learning Rate**	**Epoch**	**TC**	**MAPE**
model_0_1	1	0.0001	1	84.9%	5.9%
model_0_2	1	0.0001	5	89%	21%
model_0_3	1	0.0001	10	93.3%	36%
model_BS1_1	1	0.00001	1	83.8%	2%
model_BS1_2	1	0.00001	5	84.4%	4%
model_BS1_3	1	0.00001	10	84.9%	5%
model_BS4_1	4	0.00001	1	83.6%	1.8%
model_BS4_2	4	0.00001	5	83.8%	2.2%
model_BS4_3	4	0.00001	10	72.7%	3%
model_BS8_1	8	0.00001	1	83.6%	1.8%
model_BS8_2	8	0.00001	5	83.7%	1.9%
model_BS8_3	8	0.00001	10	83.9%	2.25%
model_BS16_1	16	0.00001	1	83.63%	1.8%
model_BS16_2	16	0.00001	5	72.59%	1.85%
model_BS16_3	16	0.00001	10	83.7%	1.96%

Table 6 This table presents the Temporal Consistency, MAPE, and edge error for different combinations of batch size, learning rate, and the number of epochs in the U-Net approach.

Chapter 6: Analysis

6.1 Post-Processing Results Analysis

Post-processing approach analysis: As is evident from the results tables, both post-processing techniques: mean kernel and exponential smoothing, have improved temporal consistency. Although with an increase in smoothing level (kernel size and alpha), temporal consistency continues to improve, it comes at the expense of MAPE and edge metrics. This compromise results in features in images becoming indistinguishable and introduces severe inaccuracies, which makes the post-processing techniques unreliable in the context of this problem.

6.2 Deep Learning Results Analysis

The results of Jacobian sparse regularization demonstrated a significant increase in temporal consistency by 4%, reaching a value of 83.22 compared to the previous 79.5%. However, this regularization had no significant effect on the MAPE and edge metrics. Among the two loss functions we tested, MSE and MAE, MAE exhibited the best performance in terms of the trade-off between TC and accuracy (MAPE and edge metrics).

The fine-tuning approach exhibited improvement in the TC metric but resulted in an unacceptable MAPE. Furthermore, as either the batch size or the number of epochs on which the model was trained increased, the TC metric improved while MAPE deteriorated, as demonstrated in Table 6.

In summary, as the model progresses through epochs, it becomes better at maintaining temporal consistency, but there is a trade-off with accuracy (MAPE), which deteriorates over time. Specifically, the model of BS equals 1 and LR equals 0.0001 at epoch 1, exhibits a decent level of temporal consistency (84.9%) with low MAPE (5.9%). However, by epoch 5, while the temporal consistency increases to 89%, the MAPE rises to 21%. This trend continues at epoch 10, with even higher temporal consistency (93.3%), but the MAPE increases significantly to 36%. These results indicate the need to strike a balance between temporal consistency and accurate data predictions when using this model.

Chapter 7: Conclusion

In conclusion, this project successfully addressed the challenge of temporal inconsistency in video depth estimation while adhering to various constraints. In its initial phase, it introduced a cutting-edge metric for quantifying temporal consistency.

The project's first approach outcome resulted in a significant 4% improvement in TC without the need for extensive network modifications or video training data. Furthermore, the innovative integration of sparse Jacobian regularization and the fine-tuning of the original model to learn consistency from the predictions of a simpler model yielded promising results, providing valuable insights for the development of more reliable systems.

To further enhance the metric, potential improvements include training on videos with consecutive frames, avoiding shuffling to promote consistency, modifying the network architecture (e.g., employing a larger U-Net model), experimenting with layer freezing and training approaches, and exploring different output options. While time limitations may have constrained the completion of these enhancements, they represent a promising avenue for future developments in this field.

References

[1] Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017, December 1). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(12), 2481–2495. https://doi.org/10.1109/tpami.2016.2644615

[2] Li, S., Luo, Y., Zhu, Y., Zhao, X., Li, Y., & Shan, Y. (2021, October). Enforcing Temporal Consistency in Video Depth Estimation. 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). https://doi.org/10.1109/iccvw54120.2021.00134

[3] Eilertsen, G., Mantiuk, R. K., & Unger, J. (2019, June). Single-Frame Regularization for Temporally Stable CNNs. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2019.01143

[4] Zhang, H., Li, Y., Cao, Y., Liu, Y., Shen, C., & Yan, Y. (2019, October). Exploiting Temporal Consistency for Real-Time Video Depth Estimation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv.2019.00181

[5] Varghese, S., Bayzidi, Y., Bar, A., Kapoor, N., Lahiri, S., Schneider, J. D., Schmidt, N., Schlicht, P., Huger, F., & Fingscheidt, T. (2020, June). Unsupervised Temporal Consistency Metric for Video Segmentation in Highly-Automated Driving. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). https://doi.org/10.1109/cvprw50498.2020.00176

[6] Lei, C., Xing, Y., & Chen, Q. (2020). Blind Video Temporal Consistency via Deep Video Prior. ArXiv, abs/2010.11838

[7] Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., & Susskind, J. M. (2021, October). Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv48922.2021.01073

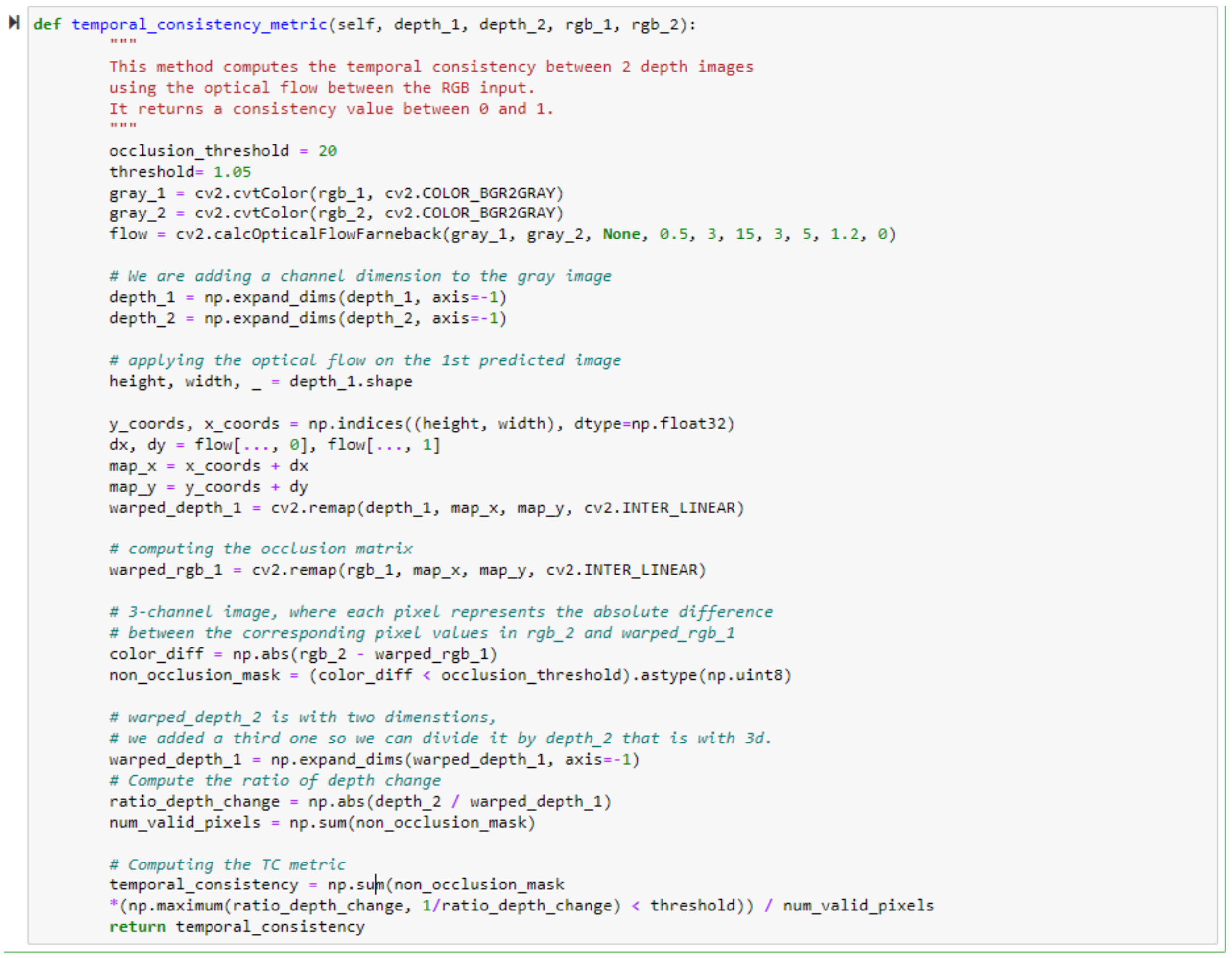
[8] Eigen, D., & Fergus, R. (2015). Predicting depth, surface normals, and semantic labels with a common multi-scale convolutional architecture. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2650-2658)

[9] Pramod, O. (2023, August 15). Loss Functions Unraveled - om pramod - Medium. Medium. https://medium.com/@ompramod9921/loss-functions-unraveled-805d76c239fe

Appendices

Appendix A

Temporal Consistency Metric Implementation



```python
def temporal_consistency_metric(self, depth_1, depth_2, rgb_1, rgb_2):
        """
        This method computes the temporal consistency between 2 depth images
        using the optical flow between the RGB input.
        It returns a consistency value between 0 and 1.
        """
        occlusion_threshold = 20
        threshold= 1.05
        gray_1 = cv2.cvtColor(rgb_1, cv2.COLOR_BGR2GRAY)
        gray_2 = cv2.cvtColor(rgb_2, cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(gray_1, gray_2, None, 0.5, 3, 15, 3, 5, 1.2, 0)

        # We are adding a channel dimension to the gray image
        depth_1 = np.expand_dims(depth_1, axis=-1)
        depth_2 = np.expand_dims(depth_2, axis=-1)

        # applying the optical flow on the 1st predicted image
        height, width, _ = depth_1.shape

        y_coords, x_coords = np.indices((height, width), dtype=np.float32)
        dx, dy = flow[..., 0], flow[..., 1]
        map_x = x_coords + dx
        map_y = y_coords + dy
        warped_depth_1 = cv2.remap(depth_1, map_x, map_y, cv2.INTER_LINEAR)

        # computing the occlusion matrix
        warped_rgb_1 = cv2.remap(rgb_1, map_x, map_y, cv2.INTER_LINEAR)

        # 3-channel image, where each pixel represents the absolute difference
        # between the corresponding pixel values in rgb_2 and warped_rgb_1
        color_diff = np.abs(rgb_2 - warped_rgb_1)
        non_occlusion_mask = (color_diff < occlusion_threshold).astype(np.uint8)

        # warped_depth_2 is with two dimenstions,
        # we added a third one so we can divide it by depth_2 that is with 3d.
        warped_depth_1 = np.expand_dims(warped_depth_1, axis=-1)
        # Compute the ratio of depth change
        ratio_depth_change = np.abs(depth_2 / warped_depth_1)
        num_valid_pixels = np.sum(non_occlusion_mask)

        # Computing the TC metric
        temporal_consistency = np.sum(non_occlusion_mask
        *(np.maximum(ratio_depth_change, 1/ratio_depth_change) < threshold)) / num_valid_pixels
        return temporal_consistency
```