Matching frame rates between IR and RGB videos with IFRNet

Utku Acar
Vestel
Manisa, Turkey
Utku.Acar@vestel.com.tr

***Abstract*—This project focuses on increasing the video frame rate from Low Frame Rate(LFR) to High Frame Rate(HFR) which helps us to address frames in videos with different frame rates for video processing compatibility. While doing so, this project works on a minimum requirement of memory too. The IFRNet model is chosen because of its lightweight computational structure compared to other models. Since IFRNet is an end-to-end model, it is easy to implement and use this model for this project without the requirement of knowing and using traditional FlowNet, Intermediate FlowNet, Approximation Refinement, and Synthesis steps separately. The focus of this interpolation is finding an intermediate frame using two consecutive frames while keeping time slots for each frame according to the new frame rate.**

***Index Terms*—Video Frame Interpolation, Image Processing, Video Processing, Deep Learning.**

I. Introduction

Video frame interpolation is one of the most popular subjects in low-level image processing since it can decrease costs very much in many cases like the elimination of the need of buying a new camera to satisfy required frame rates for videos etc. The usage of computationally heavy and expensive algorithms and software was common in the video processing area and also hardware requirements for using these algorithms were very heavy and not suitable for commercial usage [1]. In recent years, a couple of new models appeared with different trade-offs. I have chosen one of them that has an end-to-end structure and is faster than most of those algorithms as named ”IFRNet” which uses deep learning to predict intermediate frames. The end-to-end structure of this model can be seen in Fig. 1.

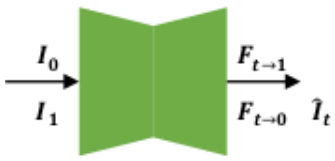

Fig. 1. IFRNet’s end-to-end Structure [2]

IFRNet has a high-gain encoding-decoding structure which is used to extract features in the context from input images via shared encoding in a pyramid-like manner. These features are then used to refine the flow of information between intermediate frames through a series of ”coarse-to-fine” decoders, resulting in the final output. In addition to the standard loss function for image reconstruction, they have also included two new loss functions: one to improve the alignment of features for synthesizing intermediate frames, and one to ensure consistency in the feature space. These loss functions are designed to guide the feature alignment process more effectively toward synthesizing intermediate frames [2]. The general structure of the IFRNet model can be seen in Fig. 2.

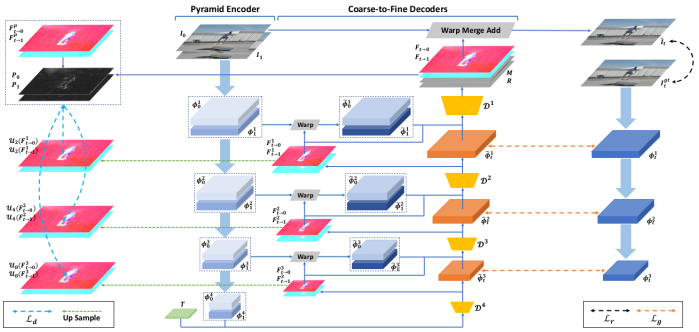

Fig. 2. IFRNet’s General Structure [2]

II. Background

A. Tools

1) Python: Python is a popular programming language that is interpreted, object-oriented, and high-level. It has a dynamic nature, which means that it is easy to use and adaptable. Its built-in data structures and support for modules and packages make it useful for developing applications quickly, as well as connecting different components together. Python’s clear syntax makes it easy to read and understand, which reduces the cost of maintaining programs. It is also free to use on all major platforms [3].

2) Anaconda: Anaconda is a distribution of the Python and R programming languages that are used for scientific computing, including data science, machine learning, and large-scale data processing. It includes a range of packages and tools for managing and deploying packages and environments and is designed to be easy to use. Anaconda is developed and maintained by the company Anaconda, Inc., which offers a number of other products, including Anaconda Team Edition and Anaconda Enterprise Edition, which are not free. The package manager in Anaconda is called Conda,

which can also be used as a standalone open-source package. There is also a small version of Anaconda called Miniconda, which includes only a limited number of packages and is designed for bootstrapping [4].

3) Pytorch: PyTorch is a machine learning framework that is open source and based on Python and Torch. Torch is an open-source library used for developing deep neural networks and is written in Lua. PyTorch is widely used in deep learning research and aims to make the transition from prototyping to deployment faster [5].

4) Shell Scripting: Shell scripts are programs that are designed to be run by a Unix shell, a command-line interpreter. There are various types of shell scripts that can be written in different shell languages. These scripts are often used to perform tasks such as manipulating files, executing programs, and printing text. A ”wrapper” is a type of script that sets up the environment, runs a specific application, and performs any necessary cleanup or logging tasks. [6].

III. The Project Work

A. Working Scheme of the Master Script

There was a need for a master script to schedule the required operations which are taken care of by python files. So I have created a shell script for this purpose. The flowchart of the script can be seen in Fig. 3.

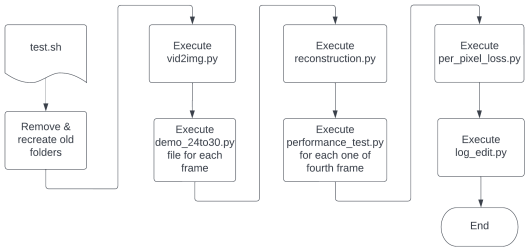

Fig. 3. Flow Chart of the test.sh

The script removes old folders and recreates them first to eliminate the residue files which could be left from earlier executions. After this initialization process, it executes a Python script to extract each frame from a given video then it executes another Python script to predict one intermediate frame from every two consecutive frames from the video which are found earlier. After this prediction phase, it executes a Python script to reconstruct the video with predicted frames in addition to actual frames then it executes another Python script to create predicted frames that match the actual frames by the time slots. After this matching operation, it executes a Python script to calculate the loss between predicted frames and actual frames and save these values to a text and create a bar plot to visualize loss values for each frame. The last part of this script executes another Python script to find the highest and the lowest loss values among the frames.

This project consists of different parts as mentioned above. Each part can be seen below:

- Extraction of frames from videos
- Intermediate frame prediction from consecutive frames
- Video Reconstruction
- Secondary frame prediction for comparison
- Calculating per pixel loss and data visualization
- Log processing

B. Extraction of Frames From Videos

This process has been made by execution of a Python script named ”vid2img.py”. It reads the video and saves each frame in the video to the target folder while there is a next frame present.

C. Intermediate Frame Prediction From Consecutive Frames

This part was also made by execution of a Python script named ”demo_24to30.py”. First, it gets the Graphical Processing Unit (GPU) information and decides which device is used by the existence of an NVIDIA GPU on the system. If there is an NVIDIA GPU on the system, then the script uses this GPU to make calculations and if there is no NVIDIA GPU on the system, it uses the CPU to make calculations. After this decision process, there is a loading process of a pre-trained IFRNet model. After selecting the pre-trained model, it reads two consecutive frames, and the decision of which frames should be read comes from the for-loop at the master bash script. After this reading of the frames, tensors are configured accordingly to predict the intermediate frame between those two frames. The inference step comes next by using these tensors to get the predicted frame’s pixel values. When these prediction has been completed, it saves these values as a png file with the name of the next frame of the second of consecutive frames chosen for each frame number. This naming is done by the current frame number, if the current frame number is 1 then save the actual first frame of the video with ”_240” postfix and if the frame number can be divisible by 4, then save both the original frame and predicted frame with different postfix (”_240” and ”_30”) and if the frame number does not satisfy these two cases which means that it cannot be divisible by 4, then save only predicted frame with ”_30” postfix.

D. Video Reconstruction

There was a requirement to recreate the video with predicted frames in addition. So this task is made by a python script named ”reconstruction.py”. It basically works like adding the frame names into an array and this addition process is bounded to a couple of if statements if the frame number equals 1 then it gets the actual first frame of the video with the ”_240” postfix and if the frame number does not equal to 1 then it gets predicted frame with ”_30” postfix and if the frame number can be divisible with 4 then add its name also with ”_240” postfix. After these addition processes in a for loop, the writing process comes next, and then it gives the output path and total frame number of the recreated video.

E. Secondary Frame Prediction for Comparison

This part is very similar to the part named "Intermediate Frame Prediction From Consecutive Frames". This process also has been made by a Python script named "performance_test.py" and the main goal is to create predicted frames to compare them with the actual frames that have the same time slot by reading each fourth and the next frame of the fourth frame and infer them and get predicted frame as an output with "_30" postfix and it also saves the actual frame with "_240" postfix.

F. Calculating Per Pixel Loss and Data Visualization

This process is made by a Python script named "per_pixel_loss.py" and starts with the initialization of the array with zeros with the size of one-fourth of the total frame number which is a loss array. After this initialization process, it reads both frames with the postfix of "_240" and "_30" for each of the fourth frames and finds the row and column numbers of the image. Then it takes the absolute differences for every pixel and all three channels and it takes the average value of these differences for all three channels. Then add these values for each pixel and save the summation to the related frame index of the loss array and also to a file named "perframeloss.txt". As the last step of this script, it plots the frame loss array as a bar plot which consists of loss values for each fourth frame.

G. Log Processing

This part has been made for processing loss values to find the frames with maximum and minimum loss values via a Python script named "log_edit.py". This script gets the file path which contains loss values and it splits the data with specific keys like square bracket, newline, and space and appends each element to an array. Then it searches through the array to find both frames with minimum and maximum loss values and it saves this info to the text file named "statistics.txt".

IV. Results & Discussions

The results of this application are just like expected both in visual and statistical ways. The predicted frames with the existence of instant movement of the camera are more blurry than the predicted frames with the movement of the camera with lesser velocity.

A. Visual Comparison of Predicted and Actual Frames with Different Pretrained Models

The visual comparison is an important element of the results. Because increasing the frame per second of the video the quality of the predicted frames should not be decreased. While we are using Go Pro pre-trained model and Lake sample video, the best-predicted frame is 64^{th} frame and the worst-predicted frame is 8^{th} frame as we can compare the blur of the images of the first two cells and last two cells from the first row of Table 1. The reflection of the trees on the water in the 64^{th} frame is sharper than the 8^{th} frame. If we change the pre-trained model to the Vimeo model the best-predicted frame stays the same as 64^{th} frame but the worst-predicted frame is changed to 132^{nd} as we can compare the blur of the images of the first two cells and last two cells from the second row of Table 1. The reflection of the trees on the water in the 64^{th} frame is sharper than the 132^{nd} frame.

While we are using Go Pro pre-trained model and Travel sample video, the best-predicted frame is 12^{nd} frame and the worst-predicted frame is 516^{th} frame as we can compare the blur of the images of the first two cells and last two cells from the third row of Table 1. The edges in the 12^{nd} frame are sharper than the 516^{th} frame and also the 516^{th} frame is over-blurry since the camera moves instantly in that frame and the black car is disintegrated. If we change the pre-trained model to the Vimeo model the best-predicted and the worst-predicted frames stay the same as 12^{nd} and 516^{th} frames as we can compare the blur of the images of the first two cells and last two cells from the fourth row of Table 1. The edges in the 12^{nd} frame are sharper than the 516^{th} frame and also the 516^{th} frame is over-blurry since the camera moves instantly at that frame and the black car integrity has been preserved.

While we are using Go Pro pre-trained model and Walking sample video, the best-predicted frame is 644^{th} frame and the worst-predicted frame is 944^{th} frame as we can compare the blur of the images of the first two cells and last two cells from the fifth row of Table 1. The edges in the 644^{th} frame are sharper than the 944^{th} frame. If we change the pre-trained model to the Vimeo model the best-predicted and the worst-predicted frames stay the same as 644^{th} and 944^{th} frames as we can compare the blur of the images of the first two cells and last two cells from the sixth row of Table 1. The edges in the 644^{th} frame are sharper than the 944^{th} frame.

The worst predicted frame of all is 516^{th} frame of the Travel sample video while Go Pro pre-trained model has been selected. The reason why this frame is the worst predicted frame is the sudden movement of the camera and loss of focus.

B. Statistical Comparison of Predicted and Actual Frames with Different Pretrained Models

The statistical comparison gives us the opportunity to analyze the data in more detail. Generally, our eyes cannot differentiate the scale of the differences in the images precisely but if we use numerical data we can see the differences in images precisely.

While we inspect Table 2, changing the Model doesn't change the frame number of the worst-predicted or best-predicted frames except for the worst-predicted frame when the Lake sample video has been used.

In general, predicted frames are better when the Vimeo Model had been chosen and the worst-predicted frames are present when the Travel sample video had been used.

If we look at the loss values of the frames in Table 2, we can see that 64^{th} frame is the best-predicted frame when the Vimeo model and Lake sample video had been chosen and the worst-predicted frame is 516^{th} frame when the Go Pro Model and Travel sample video had been chosen.

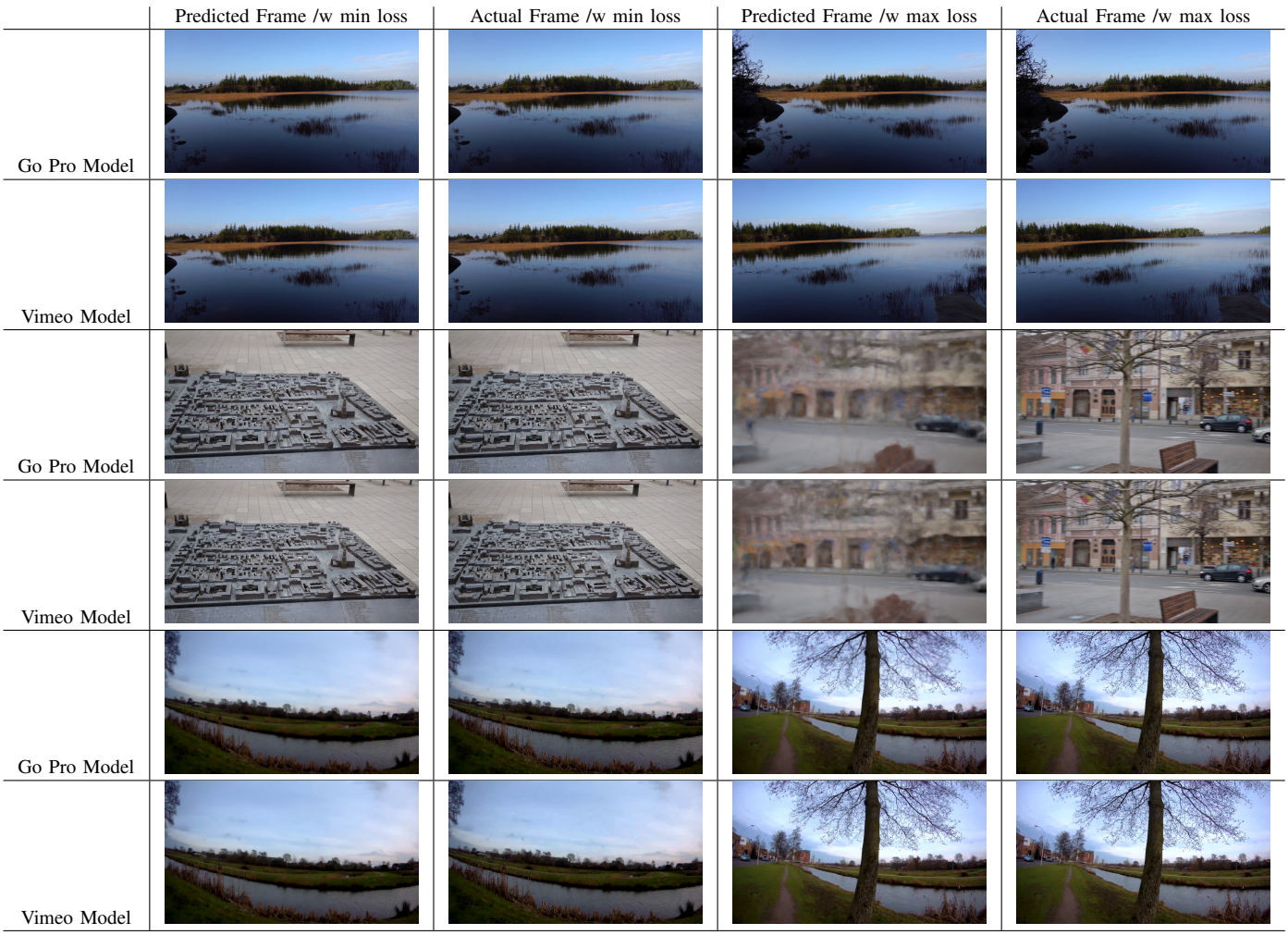

TABLE I
FRAMES WITH MINIMUM AND MAXIMUM LOSSES

Sample Name	Pretrained Model Name	Min/Max Loss	Frame Number	Loss value (in percent)
Lake	Go Pro	Min	64	0.589
Lake	Go Pro	Max	8	0.968
Lake	Vimeo	Min	64	0.512
Lake	Vimeo	Max	132	0.724
Travel	Go Pro	Min	12	0.899
Travel	Go Pro	Max	516	7.617
Travel	Vimeo	Min	12	0.636
Travel	Vimeo	Max	516	7.537
Walking	Go Pro	Min	644	0.859
Walking	Go Pro	Max	944	5.084
Walking	Vimeo	Min	644	0.716
Walking	Vimeo	Max	944	5.277

TABLE II
LOSS VALUES FOR DIFFERENT MODELS AND VIDEO SAMPLES

V. RELATED WORK

Video frame interpolation with the help of Deep Learning is a very popular topic in computer vision nowadays. There are different algorithms with different approaches such as kernel-based, flow-based, and phase-based. Some of them use motion flow information to predict intermediate frames just like ”IFRNet” while some of them use kernel information instead of motion flow just like ”VFI via Adaptive Convolution”. Also, there are some algorithms that use both kernel and motion flow information between images to make predictions well-balanced just like ”AdaCoF”. One example

of a model for phase-based prediction is ”PhaseNet”.

Video Frame Interpolation via Adaptive Convolution: This project does not use motion flow vectors unlike our project (IFRNet), it uses a kernel-based approach instead. It convolves patches in the images and gets a spatially elastic kernel using CNN. The usage of a kernel-based approach saves them from the division of the interpolation into the steps like motion flow. They are also using as their say ”edge-aware convolution kernel” to save the edge pixel values of predicted frame [7].

AdaCoF: Adaptive Collaboration of Flows for Video Frame Interpolation: AdaCoF is a project which is inspired by the DefConv structure. This project uses the motion flow method just like ours (IFRNet) and it also uses a kernel-based method just like ”VFI via Adaptive Convolution”. AdaCoF also differs in the number of flow vectors they store in order to get pixel-level predictions. IFRNet uses just 1 motion flow vector for each frame while the AdaCoF uses much more than 1 motion flow vector for each pixel and also samples according to their reference locations to reduce the possibility of distortion when there are sudden angle changes or loss of focus etc present. It also processes these sampled flow vectors to predict the correct intermediate frame's pixel value. It's a computationally expensive way to predict each pixel [8].

PhaseNet for Video Frame Interpolation: This project uses a different approach named ”Phase-based method” which is completely different from our algorithm (IFRNet). It reverts the order of the phase decomposition and gets it as an input to predict both phase and amplitude values for different levels and the predicted frames generated by using those values. This PhaseNet Structure also shares weights between channels and pyramid levels just like our project (IFRNet). This project is more computationally heavy if we compare it with IFRNet. [9].

VI. Conclusion

Since the goal was to increase the frame number of 24-frame video to the 30-frame video while preserving the image quality and lowering the computational requirements, it has been done in a way that works on even low-end NVIDIA GPUs with lesser Video RAM than usual. Also, there was a motion blur and distortion present in predicted frames when the camera angle changes rapidly which resulted in higher loss values. Predicted frames with the Vimeo Model are more successful than predicted frames with Go Pro Model in case of loss values and predicted frame context integrity.

References

[1] “Large motion frame interpolation,” – Google AI Blog. [Online]. Available: https://ai.googleblog.com/2022/10/large-motion-frame-interpolation.html. [Accessed: 09-Jan-2023].

[2] L. Kong, B. Jiang, D. Luo, W. Chu, X. Huang, Y. Tai, C. Wang, and J. Yang, “IFRNet: Intermediate Feature Refine Network for efficient frame interpolation,” arXiv.org, 29-May-2022. [Online]. Available: https://arxiv.org/abs/2205.14620. [Accessed: 09-Jan-2023].

[3] “What is python? executive summary,” Python.org. [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 12-Jan-2023].

[4] “Anaconda (python distribution),” Wikipedia, 19-Oct-2022. [Online]. Available: https://en.wikipedia.org/wiki/Anaconda_(Python_distribution). [Accessed: 12-Jan-2023].

[5] K. Yasar and S. Lewis, “What is pytorch?,” Enterprise AI, 16-Nov-2022. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/PyTorch. [Accessed: 12-Jan-2023].

[6] “Shellscript,” Wikipedia, 16-Jan-2008. [Online]. Available: https://en.wikipedia.org/wiki/Shellscript. [Accessed: 19-Dec-2022].

[7] S. Niklaus, L. Mai, and F. Liu, “Video frame interpolation via Adaptive Convolution,” arXiv.org, 22-Mar-2017. [Online]. Available: https://arxiv.org/abs/1703.07514. [Accessed: 12-Jan-2023].

[8] H. Lee, T. Kim, T.-young Chung, D. Pak, Y. Ban, and S. Lee, “ADACOF: Adaptive collaboration of flows for video frame interpolation,” arXiv.org, 08-Mar-2020. [Online]. Available: https://arxiv.org/abs/1907.10244. [Accessed: 12-Jan-2023].

[9] S. Meyer, A. Djelouah, B. McWilliams, A. Sorkine-Hornung, M. Gross, and C. Schroers, “Phasenet for Video Frame Interpolation,” arXiv.org, 03-Apr-2018. [Online]. Available: https://arxiv.org/abs/1804.00884. [Accessed: 12-Jan-2023].