Video Frame Interpolation using Pyramid Representation

Jan Murić, Paulo Erak
CISUC - Centre for Informatics and Systems of the University of Coimbra ; University of Coimbra, Portugal
{uc2024189891, uc2024164543}@student.uc.pt

Abstract

The primary goal of this paper is synthesizing non-existent intermediate frames between given input frames. By achieving this, we can enhance video quality in several ways, such as increasing frame rates, creating slow-motion effects, improving video compression, and generating animations from fewer frames. One of the possible ways to achieve this is through calculation and use of optical flow. This paper focuses on estimating optical flows from different levels of feature pyramid representation of our inputs. We then combine them, use them for feature warping and in final construction of an intermediate frame. We use convolutional models for feature extraction, flow estimation, creation of the intermediate frame and its refinement. Through this approach we try to minimize artifacts and improve the visual quality of synthesized frames. We provide insight into related work that inspired this paper, reflect on problems encountered during development and possible improvements that can be made to our model. Source codes are available at: https://github.com/MuricJ/AI_video_interpolator.git

1 Introduction

Video frame interpolation is a form of video processing which aims to generate intermediate frames between two given input frames. By achieving this, we can enhance video quality in several ways, such as increasing frame rates, creating slow-motion effects, improving video compression, and generating animations from fewer frames. Video frame interpolation has significant applications in entertainment, media editing, and machine learning tasks, making it an essential research area in computer vision.

The area of video frame interpolation is a very active field with a fair number of important articles being written in the last eight years. Articles specialize in a wide range of improvements: focus on smaller and faster-moving objects (Wu, 2023), frame interpolation for large movements (Reda, 2022), unsupervised learning for use of smaller datasets (Simon Meister, 2017), lightweight models (Hahm, 2023), use of blurry input frames (W. Shen, 2020), etc. This range shows that there are many ways in which the field can advance, and that future work seems promising. There is also a healthy split between advancement of well-established ideas and concepts such as optical flow (Hahm, 2023) (Simon Meister, 2017) (Reda, 2022) and moving away from them in search of newer, alternative methods (Ziwei Liu, 2017).

During our exploration of the topic, and subsequent reading of research papers on the matter, we found the well-established idea of optical flow intriguing and quite intuitive. Optical flow being the distribution of the apparent velocities of objects in an image. By estimating optical flow between video frames, we can measure the velocities of objects in the video. This gives us an idea of the location of the object in the next frame. This concept seems intuitive and easy to follow. The truth is, however, that optical flow between two frames is often hard to properly calculate.

Several research papers describe unique methods for addressing challenges connected to optical flow estimation. We found the method used in FILM (Reda, 2022) particularly intriguing and inspiring. The method used in FILM is made to combat blurriness that usually comes with large motions between frames. FILM method uses a pyramidal representation of input frames (image pyramids) where each layer represents smaller and smaller resolutions of input frames. This provides us with a range of scales which offer a coarser or finer look into the frames. Features are extracted at each pyramid level using a specially made feature extractor that shares weights between levels, turning the image pyramids into feature pyramids. Starting from the coarsest level of feature pyramids, optical flow is estimated, upscaled and sent to the next layer for further optical flow refinement. The feature pyramid is then warped on each level using the optical flow estimated on that level. Warped features and optical flows are jointly sent to the decoder for fusion and refinement producing an intermediate frame. Having the method's original goal in mind, this layered approach to the problem makes sense. Particularly because large movement can be seen and addressed on lower resolutions and then further refined on higher resolutions where more details become apparent. Extreme motions still prove to cause problems causing un-natural deformations. Using this method as an inspiration we strive to make our model achieve visually pleasing intermediate frames.

We propose a similar model with changes to the warping procedure and inputs into the decoder. We assume that by using inversed flow on extracted features in the first part of training we can make our model more robust to subtle differences between optical flows from one frame to another. We also send the original non-warped features into the decoder to give it more context on unmodified data. Other modifications, including the dampening of optical flows, loss functions, and differences in the resources available for training and testing our model compared to those in the original inspirational work (Reda, 2022), are discussed in more detail later in the paper.

In the remainder of this paper, we will discuss related work in greater detail, describe the materials used and their sources, explain the methods employed along with the rationale behind their use and their implementation, present the experimental setups and results, and conclude with our final thoughts on the project.

2 Related Work

Video frame interpolation is a long-standing idea that has inspired different approaches to its execution.

Pyramidal representation is presented as a good starting point for image processing tasks (Hahm, 2023). In pyramidal representation an image is subjected to repeated smoothing and subsampling. Lowpass pyramidal representation generates a pyramidal representation of an image where each layer presents that image at different resolutions. Lower resolutions provide a coarser overview, and higher resolutions provide finer overview. This should work in favour of larger motions (Reda, 2022) because larger motions can be detected at lower resolutions and subsequently refined on finer levels of the pyramid. Using original pixels from input images and going forward through any kind of model would be cumbersome. Instead of using pixels, features are extracted on each level of the pyramidal representation. Feature extraction is carried out using convolutional layers. Extracted features are used in later calculations and image synthesis (Hahm, 2023) (Reda, 2022). The result of extracting features on each level of the image pyramid is the feature pyramid.

One of the more prominent approaches to video frame interpolation uses optical flow, a concept that stems from psychology (Gibson, 1950). Optical flow is described in literature as the distribution of the apparent velocities of objects in an image or in our context between images. Displacement vectors are calculated between two or more input images and represent the direction and amount of amount of movement of objects between frames. Provided the correct estimation of optical flow, we can, in theory, anticipate positions of objects in moments between frames using a multiplication factor between 0 and 1. Calculating optical flow is a non-trivial matter and many papers focus on refining calculations (W. Shen, 2020) (Liu, 2020) (Hahm, 2023) or finding ways around problems connected to the calculations (Ziwei Liu, 2017) (Simon Meister, 2017) (Reda, 2022). Bidirectional flow estimation is often considered as a good practice (Simon Meister, 2017) (Liu, 2020). Bidirectional flow considers an optical flow from the first input frame to the second input frame and an optical flow from the second input frame to the first input frame. "*The advantage of using bidirectional optical flow is that the extracted features are screened twice, which ensures the quality of the features and prevents the subsequent work from being affected by the quality of the features."* (Jiang, 2023). If pyramidal representation is used, the optical flow can be calculated for each level of the pyramid.

In the situation where the proper optical flows are obtained, they can be used in the process of warping input images. Warping requires a specially built function and can't be easily understood by a neural network. This is why warping stands by itself. Warped images usually do not provide a viable intermediate frame. Combinations of warped images give clues to the model in what way the final intermediate frame should be constructed.

Decoder stands at the end of the model, and it is responsible for intermediate image synthesis. The decoder will be fed information that was calculated throughout earlier parts of the video interpolation process. Depending on the method the decoder could be fed warped frames, initial estimate of intermediate frame, original input frames, scaled bidirectional flow and warped context features (Hahm, 2023), warped features and optical flows (Reda, 2022), warped input frames and feature pyramids (Liu, 2020), etc. The general rule is that we need to provide enough information to the decoder for it to be able to construct a viable intermediate frame. The quantity and quality of the information we feed the decoder impacts results greatly.

The main model that served as our inspiration is the model used in FILM: Frame Interpolation for Large Motion (Reda, 2022). The model and reasoning behind it intrigued us and gave us ideas for development of our version of the model. FILM model takes two frames as inputs and makes a pyramidal representation of them. Each layer of the image pyramid is subjected to feature extraction using a UNet-style encoder that allows weight sharing across the scales. Each scale has its features extracted at multiple levels of depth. Features extracted on level *l+1* are pulled through the average pool layer of stride 2 before being processed on level *l*. More features are extracted at deeper levels; UNet encoders dedicated to deeper depths have a greater number of output channels than the encoders at shallower levels. UNet encoders that are on the same depth level share weights between scales. The last part of feature extraction is concatenation of feature maps that share spatial dimensions but are of different depth levels. This process creates a "scale-agnostic" feature pyramid.

With the extracted feature pyramids FILM model continues to the next stage, calculation of a bidirectional motion at each pyramid level. The process of bidirectional motion calculation starts from the coarsest level. Model predicts task-oriented flows, $Wt\rightarrow0$ and $Wt\rightarrow1$, from the intermediate frame to the input frames. Task oriented flow is computed at each level as the sum of the upsampled flow from the coarser level *l+1* of the feature pyramid and predicted residual. Predicted residual is calculated by the stack of convolutional layers. The stack of convolutional layers is fed concatenation of original features and warped features. *Original features* meaning features of one of the input frames extracted at that level. *Warped features* are features of the other input frame that have been warped to resemble *original features*. The convolutional stack learns to output residual flows which supplement the upsampled flow and through their summation we obtain optical flows for level l. *"This process is based on the intuition that large motion at finer scales should be the same as small motion at coarser scales."* (Reda, 2022) Our model follows these same principles and methods with one exception. Our stack of convolutional layers use dampers which reduce the residuals for a constant factor. Dampers increases stability and consequently speed up training of the model.

After obtaining the oriented flows, FILM model creates backward warped feature maps of both frames on each level.

Warped features and task-oriented flows are concatenated at each level and sent to the UNet-like decoder for the final part called Fusion. In Fusion, information from coarser level *l+1* is upsampled and concatenated to the input information from level *l*. Joint information from levels *l* and *l+1* is fed to the stack of convolutional layers. Output from the stack of convolutional layers is propagated to the finer level *l-1*. This process repeats until model reaches the finest level. Output from the finest level represents the final version of intermediate frame.

This paper tries to recreate and improve the FILM model and therefore it is assumed that the reader is familiar with FILM's working principles. We implement changes in feature extraction, flow estimation, warping of input frames and change inputs into the decoder for frame synthesis. Changes are also made to hyperparameters and contributions of L1, perceptual and style loss to total loss.

3 Methods

In this section we will describe our model and the key differences from FILM, as well as the reasoning behind the changes. It takes two inputs images x_1 and x_2 and tries to predict the output image y.

The model starts with a hierarchical feature extractor that aims to extract features in a scale-agnostic manner. In the original FILM model this is achieved using weight sharing.

Let $D(x)$ and $U(x)$ be the 2x downsample and upsample operators (we use max pooling and bilinear interpolation, respectively). For both input frames $x \in \{x_1, x_2\}$ we calculate $D_n(x) = D(D_{n-1}(x))$ and $U_n(x) = U(U_{n-1}(x))$ where $D_0(x) = U_0(x) = x$.

Let $\mathrm{conv}_n(\text{x,in,out})$ be a function that applies a convolutional block with *in* input channels and *out* output channels to the input x. The first level of features is extracted as follows

$$f_{1i} = \mathrm{conv}_i(\text{x,in\_channels,B}) \text{ for all } \forall i \in \{0 \dots 4\}$$

This stands in contrast to to FILM as it strays away from the assumption that the feature extractor should share weights on all pyramid levels. In our testing we found that the Unet-style decoder struggles to learn the identity with FILM's encoder. To combat this issue, the first layer of the encoder is not constrained by the lower downsampled layers. Subsequent feature extractor layers follow a weight sharing architecture and can be expressed as follows in the second layer:

$$f_{2i} = \mathrm{conv}_5(D(f_{1i}), B, 2B) \forall i \in \{0 \dots 3\} \quad (1)$$

and in the third layer:

$$f_{3i} = \mathrm{conv}_6(D(f_{2i}), 2B, 4B) \forall i \in \{0 \dots 2\} \quad (2)$$

Finally, to get the final features, masks from layers with equal spatial dimensions are concatenated, as we are assuming that features are size-agnostic:

$$\begin{gathered} P1 = f_{10} \\ P2 = \mathrm{concat}(f_{20}, f_{11}) \\ P3 = \mathrm{concat}(f_{30}, f_{21}, f_{12}) \\ P4 = \mathrm{concat}(f_{31}, f_{22}, f_{13}) \\ P5 = \mathrm{concat}(f_{32}, f_{23}, f_{14}) \end{gathered} \quad (3)$$

Observe that increasing i is equivalent to going deeper in the feature extractor, where each layer is pooled and has the spatial dimension reduced by two, whereas increasing j is equivalent to starting the feature extraction with an image which has been downsampled by a factor of two j times. Therefore, it's easy to see that concatenating the channels in the above proposed fashion is valid dimension-wise, as the sums of the indexes of every feature f_ij in one level P_k is always equal the feature level $k = i + j$. $P(x) = (P1, P2, P3, P4, P5)$ where $P(x)$ is called the feature pyramid of the image x.

In the first step of our model, we calculate the features $P(x_1)$ and $P(x_2)$ of both of the input frames and pass them to second part of the model – the flow accumulator. The feature pyramids extracted from the encoder are used to compute the optical flow between the two input frames using residual accumulation in exactly the same fashion as the FILM model. Let

$$F_k(Pi, Pj) = \sum r\,(Pi_m, Pj_m) \quad (4)$$

and

$$F(Pi, Pj) = (F5, F4, F3, F2, F1) \quad (5)$$

The optical flow in FILM is expressed as an absolute pixel displacement in x and y dimensions, however we normalize the flow by dividing it with image dimensions $s = (v, w)$. In this way, a flow vector $v = (a, b)$ would be equivalent to a pixel displacement of length $\sqrt{((av)^2 + (wb)^2)}$. We observe that the maximal possible norm $\|\mathbf{v}\|$ is equal to $\sqrt{2}$ which helps keep flow values from exploding or causing errors with numerical precision. In a 1000x1000 static image with a maximal displacement of 5% (or 50 pixels) the largest flow in pixel terms is 50 and the smallest is 0 (static part of the image), however in the same image the difference of normalized flows is only 0.05. Furthermore, having a normalized flow value is beneficial in the context of a weight-sharing pyramid representation. As the flow is upscaled in the residual calculation, it must be multiplied by the upscale factor to maintain consistency, as its value is dependent on the image resolution. This may also pose a problem for the convolutional block responsible for calculating the flow residual, as it must learn to calculate the flow residual that is nominally different at different pyramid levels, but still corresponds to the same spatial displacement.

In trying to stabilize the learning process with normalized optical flow, we also propose to use a damping factor d_f and define the dampened flow as:

$$F'^{(P_i,P_j)} = \left(d_f F_5, d_f F_4, d_f F_3, d_f F_2, d_f F_1\right) \qquad (6)$$

where d_f is a number such that in the initial stage of training F' is close to 0. In the case of Xavier initialization (Glorot, 2010) which we used, $d_f = 0.01$ was sufficent. The idea behind this modification is simple: in the initial stages of training we assert that the zero function is a good approximation for the flow. Since we are dealing with frame interpolation of natural videos, the difference between two frames is expected to be on the order of 1/60 to 1/24 seconds, where we do not expect to find large visual displacements between the two frames. Instead of starting with a large, normalized flow value that may, in the initial stages of training, distort the image beyond recognition, we let the warp stay small and allow it to move out of the approximate solution as the training progresses.

After calculating $P(x_1)$ and $P(x_2)$ and the dampened flows in both directions $F_{12} = F'\big(P(x_1), P(x_2)\big)$, $F_{21} = F'\big(P(x_2), P(x_1)\big)$, where F_{12} represents the flow that needs to be applied to x_1 to transform into x_2 and F_{21} represents the flow that needs to be applied to x_2 to transform it into x_1, we apply the warp as follows:

$$\begin{aligned} P1_w &= \mathrm{warp}(P(x_1), -0.5F_{21}) \\ P2_w &= \mathrm{warp}(P(x_2), -0.5F_{12}) \end{aligned} \qquad (7)$$

Where warp() applies a bilinear resample warp to every level of the feature pyramid with their respective flow approximations and constructs a new warped feature pyramid.

Note that the above transformation implicitly assumes that

$$F_{12} = -F_{21} \qquad (8)$$

because under that assumption

$$\begin{aligned} P1_w &= \mathrm{warp}(P(x_1), 0.5F_{12}) \\ P2_w &= \mathrm{warp}(P(x_2), 0.5F_{21}) \end{aligned} \qquad (9)$$

which is the unconstrained transformation. We use the former method of the warp transformation in the initial stages of training to constrain the possible values of F_{12} and F_{21}. With this constraint the network quickly learns to superimpose the features over each other before passing them to the decoder. While this constraint limits the expressive power of the network to linear motion, it can be easily removed after a sufficiently good flow estimator has been learned by replacing the first warping regime with the second on an already trained network and continuing the training. This is valid because under the assumption (8), both warp regimes are are equivalent, but the second one removes the constraint.

The last part of our model is a decoder. In the original FILM model, the decoder is a Unet-style decoder that takes only the warped features and the flow estimates to reconstruct the original image. We found that this decoding scheme is too reliant on a very accurate flow estimator, as there is no information flow from the original unwarped image to the decoder. To allow the reconstruction of the original features, we concatenate the original feature pyramid to the decoder input, as well as the warped feature pyramid together with the flow estimates. As such, on every level i of our decoder, the input is:

$$\begin{gathered} \mathrm{dec}(i) = \mathrm{conv}(\mathrm{concat}(P1_i, P2_i, \\ P1i_w, P2i_w, \\ F_{12i}, F_{21i}, \\ U(\mathrm{dec}(i-1)))) \end{gathered} \qquad (10)$$

where $\mathrm{dec}(0)$ is an empty tensor and $\mathrm{out} = \mathrm{conv}\big(\mathrm{dec}(5)\big)$ is the final output of our model. The convolutional channels half in every step and at the final step the output is finally passed through the last convolutional layer with kernel size 1 to produce the RGB channels.

4 Materials

We used the Vimeo90K (Xue, 2019) to train our model and we used the factor B=24 for the base number of channels, which means the model has 33.4M parameters. We trained the model on 256x256 image crops with the Adam optimizer at batch size 7 using a learning rate of 1e-4 and reducing it after 4 epochs to 5e-5 and then training for 4 more epochs. After this, the model was trained for 1 epoch on full-resolution Vimeo90K for better noise resilience. As it’s a fully convolutional model, it’s resolution-independent. The model was implemented in Pytorch and trained on a single laptop RTX 4060 GPU.

We minimized L1 loss and when the result was satisfactory we switched to the style loss proposed in the FILM paper.

5 Results and discussion

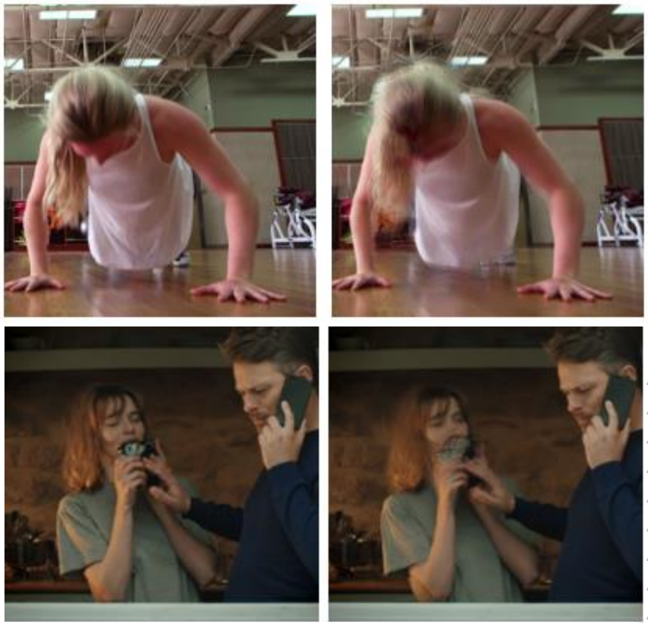

Figure 1: On the left are the original intermediate frames and on the right are the frames generated by our model using L1 loss.

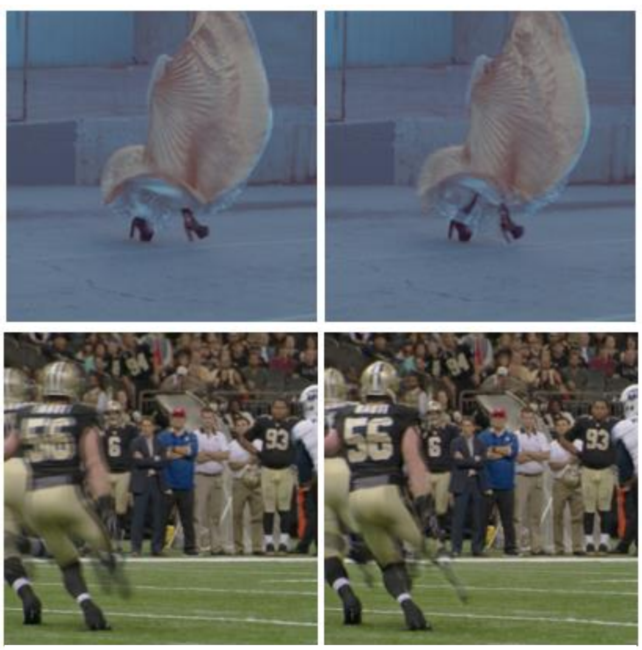

Figure 2: On the left are the original intermediate frames and on the right are the frames generated by our model using L1 and style loss.

We trained the network with L1 for 8 epochs, which is in total only ~50K iterations at our batch size, compared to FILM which was trained for 3M iterations. Due to computational constraints, training at a high resolution or with a large batch size wasn't feasible.

We found that the L1 model gave good results for video frames that do not exhibit large motion. In frames that exhibited large motion, the model would produce blurry images. This is a known limitation of using L1 loss.

	Vimeo90K
	PSNR (mean)
Our model L1 loss	33.0361
Our model L1 + style loss	33.3934

Table 1: Comparison of PSNR for two iterations of our model on Vimeo90k dataset.

Using style loss, the model produces sharper images with fewer blurry regions, even with motion that would have otherwise caused blurring. While the intermediate frames did look better when viewed as stills, we found that using style loss for the purpose of video interpolation has multiple problems. For one, minimizing style loss makes the model more likely to produce outputs that are very similar copies of one of the input frames. This causes interpolated videos to look choppy, because the intermediate frame does not really represent a frame that is temporally in between the two input frames. Furthermore, we found that artefacts produced by models minimized with style loss tend to be more visually intrusive as they tend to have sharp edges and high contrast, whereas most of L1 artefacts were happening strictly in regions with high motion and were typically blurry – in a way resembling motion blur that naturally occurs when recording fast motion with a high exposure time. Because of this, we consider L1 to be superior when interlacing original video frames with generated ones.

7 Conclusion

In this paper we gave insights into the topic of Video Frame Interpolation and the current state of the field, as well as possible ways in which the field can advance. We summarized and discussed key concepts and past works that have been crucial to our project. We explored ways of modifying existing methods of video frame interpolation, specifically FILM model (Reda, 2022). We presented our modifications as well as reasonings behind them. We presented and discussed our results obtained on our model using two versions of a loss function. We found that we were able to train a model for only 8 epochs with damper and normalized flow and got visually pleasing results. Further work on training the model with higher computational power is necessary to find the performance ceiling of our model.

8 References

Gibson, J. J. (1950). *The perception of the visual world.* Houghton Mifflin.

Glorot, X. a. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249--256.

Hahm, X. J.-h. (2023). A Unified Pyramid Recurrent Network for Video Frame Interpolation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1578-1587. Retrieved from https://arxiv.org/abs/2211.03456

Jiang, H. &. (2023). Bi-directional Optical Flow-based Feature Point Tracking Method. *Journal of Physics: Conference Series. 2477*.

Liu, S. N. (2020). Softmax Splatting for Video Frame Interpolation. *CoRR, abs/2003.05534*, 10. Retrieved from https://arxiv.org/abs/2003.05534

Meister, S. a. (2018). UnFlow: Unsupervised Learning of Optical Flow With a Bidirectional Census Loss. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/12276

Reda, F. K. (2022). FILM: Frame Interpolation for Large Motion. *Computer Vision -- ECCV 2022*, 250--266. Retrieved from https://arxiv.org/abs/2202.04901

Simon Meister, J. H. (2017). UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. *CoRR*, *abs/1711.07837*. Retrieved from http://arxiv.org/abs/1711.07837

W. Shen, W. B. (2020). Blurry Video Frame Interpolation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10.

Wu, C. Z. (2023). Video Frame Interpolation with Densely Queried Bilateral Correlation. *ArXiv, @article{Zhou2023VideoFI,*. Retrieved from https://arxiv.org/abs/2304.13596

Xue, T. a. (2019). Video Enhancement with Task-Oriented Flow. *International Journal of Computer Vision (IJCV)*, *127*, 1106--1125.

Ziwei Liu, R. A. (2017). Video Frame Synthesis using Deep Voxel Flow. *CoRR, abs/1702.02463*. Retrieved from http://arxiv.org/abs/1702.02463