Consistent Depth Estimation for Video

by
Adam Pikielny

A Thesis submitted in partial fulfillment of the requirements for Honors
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2022

This thesis by Adam Pikielny is accepted in its present form by
the Department of Computer Science as satisfying the research requirement
for the awardment of Honors.

Date _____ _____

James Tompkin, Reader

Date _____ _____

Srinath Sridhar, Reader

Acknowledgements

This work was completed in collaboration with Marc Mapeke and James Tompkin. Thank you to Marc for his persistence and attention to detail. None of these experiments would have been completed without him. Thank you to Professor Tompkin for mentoring me throughout my time at Brown and teaching me how to be a diligent, curious, and kind researcher. Thank you to Numair Khan and Yiqing Liang for offering ideas and code. Thank you to Srinath Sridhar for being the second reader of this work and providing valuable insight. Dedicated to Mom, Dad, and Noah.

Contents

Chapter 1

Introduction

1.1 Background

Depth estimation, the problem of estimating per-pixel depth of an image, is central in computational photography. Tracking, object insertion, and masking all make use of an image's depth. As shown in Figure 1.2, many smartphone cameras blur parts of the scene proportionally to their estimated depth, creating synthetic depth of field. This thesis focuses on depth estimation in video. Although monocular depth estimation is viewed as a solved problem for still frames, it is generally not consumer ready in the video case. Extending depth estimation to video will enable a similar expansion of applications for computational videography, especially in the mobile space.

Until recently, depth estimation research used pairs of images taken from different poses as input. Similarly to the human eye, one can measure the distance between corresponding points in two images to calculate relative depth. This method assumes that points have not moved between the frames. If the two inputs were captured at different points in time, this requires a static scene. Even in the case of synchronized stereo input, this method still requires highly textured surfaces to be able to accurately determine what is the same point in two images. Put simply, it is difficult to pinpoint the same part of a blank wall in two images. Without knowledge of the camera system

Figure 1.1: An example image and corresponding depth map

(a) Original Image (b) Synthetically Blurred Image

Figure 1.2: The iPhone's Portrait Mode uses synthetic depth of field, made possible by depth estimation

being used or a known object for calibration, we cannot acquire metric/absolute depth. However, relative depth is sufficient for most photography applications. An example depth map is shown in Figure 1.1. We visualize our results using inverse depth maps, where objects at a greater distance from the camera appear darker.

Recently, state of the art deep learning methods have yielded impressive results using only a single image as input. These neural monocular depth estimation methods remove the need for establishing point correspondences and estimating poses. This allows depth estimation to be widely used without a pair of stereo images, along with eliminating sources of error.

1.2 Problem Illustration and Proposed Solution

While monocular depth estimation methods are widely used in photography applications, their output on videos is not consumer ready. The central problem is establishing temporal consistency in depth maps between successive video frames. One approach is to solve an optimization problem or attempt to fine-tune the model's weights at inference time, in order to establish consistency [1]. In addition to being slow, these processes relies on point correspondences, which in turn limit applications to static scenes. Theoretically, this post-processing shouldn't be necessary if the model initially learns a properly smooth function. However, methods that only estimate each input frame individually exhibit significant fluctuation between outputs. This causes flickering and artifacts when viewed as a video. This problem exists even in scenes with little or no motion, where consecutive inputs look nearly identical, but outputs do not.

Using the state of the art work of Li et al. [2], we were able to demonstrate shortcomings of current systems. The simplest case, we reason, is a video with essentially no camera or subject movement. In this instance, there is no reason that the model should exhibit temporal inconsistencies in depth. Without moving objects, we would expect depth to be identical between frames. We

collected several handheld videos with varying motion. First we collect a static video, where the only motion is introduced by hand shake. We next collected videos that introduce subtle lateral and rotational subpixel movement between frames. In these simple cases, the model of Li et al. [2] exhibits ubiquitous inconsistencies. When examined individually, each output appears to accurately estimate depth. Played as a video, however, we see abundant flickering as certain objects appear to jump forward or backward in depth between frames.

This allows us to define our problem. Even in the simple case of minute pixel or subpixel movements between frames, the depth estimation of state of the art models still does not exhibit temporal consistency. Surely if this is true, any videos with larger motion will suffer from the same problem. Any application of these depth maps, such as synthetic depth of field or object segmentation, will suffer from flickering in the output video.

Our goal is to create a model that will output consistent depth maps at inference time, without any post-processing requiring optimization among many frames. In addition to being fast, this generalizes better to scenes with motion, as it does not rely on point correspondences or enforce any assumptions about the scene or camera. Moreover, it is theoretically possible to create such a network; as such, we believe this is a stronger research direction to pursue compared to hackier solutions applied afterwards.

Upon investigation, we find that current state of the art works struggle to successfully map near-identical inputs to consistent latent encodings. These models encode overly high frequency signals, which cause inconsistency. In this work, we explore several techniques to learn a smoother depth estimation model and latent encoding. First, we explore regularization losses to constrain the model to learn a smoother bottleneck encoding. This technique successfully improves consistency, however it reduces the model's ability to accurately predict depth, especially in high frequency areas. Second, we theorize that aliasing problems are the root of our inconsistency, and therefore attempt to modify existing models to be more translation and rotation equivariant. Using improved sampling methods, Fourier features, and 1x1 convolution kernels, we are able to improve consistency, but still have issues retaining accuracy of highly detailed scenes.

We believe that techniques presented in this work can improve many neural depth estimation model architectures. Furthermore, stereo methods frequently use a neural refinement or preprocess because it is faster, so any techniques presented here could also help with the more well defined problem setting of multiple input images. Although we focus on scenes that are static (both in camera pose and scene objects), our work generalizes to dynamic scenes, and we believe can impact myriad neural videography networks that require consistency between frames.

Chapter 2

Related Work

2.1 MannequinChallenge

Li et al.'s "Learning the Depths of Moving People by Watching Frozen People" [2] focuses on depth estimation of dynamic scenes involving humans. This work uses YouTube videos from the Mannequin Challenge –in which people attempt to remain motionless (imitating mannequins), typically filmed with a dynamic camera– as a dataset to estimate depth, using the assumption that these scenes involve a dynamic camera but a static scene. The authors provide their dataset, which includes videos along with associated camera poses and intrinsics, collected using ORB-SLAM2 [3] and refined with an SfM system [4]. Their best model involves an input mask of humans in the scene, and generates depth for humans and static backgrounds separately. They have a simpler model that takes in only an image as input. We use this work's dataset to generate training data for our model, and we use their trained models and Autoencoder architecture, based on [5], as a baseline.

2.2 Consistent Video Depth

The goal Luo et al.'s work[1] is to reconstruct geometrically consistent dense depth with no flickering through time. This technique randomly samples pairs of (non-consecutive) frames, estimates their depths individually, then fine tunes the network for each frame to establish consistency between frames. Consistency is established by finding point correspondences using flow. Results of this paper are good; however the fine-tuning process is extremely time and computationally intensive, which is not realistic for online computational photography use cases. It certainly would not work at the time of capture, for example, looking at live footage with a synthetic blur applied. Also, this method relies on finding static point correspondences, enforcing the assumption that the filmed scene has static subjects.

2.3 Alias-Free Generative Adversarial Networks

The goal of Karras et al.'s paper [6] is to introduce translational and rotational equivariance into GAN generated human faces. While this work is not related to depth estimation, the principles the authors use to establish consistency between frames with subtle translation can be applied to our problem. Mathematically, they enforce $g[t(z_0)] = t[g(z_0)]$, where t is the translation, $g()$ is a generator, and z_0 is an input vector (which can be interchanged with a monocular image, for our purposes). Put simply, translating a generated face image should look equivalent to translating the input vector before generating the face. The authors expose aliasing issues commonly found in GANs, specifically in their own StyleGAN2 [7]. Specifically, the paper discusses texture sticking, where high frequency textures (such as facial hair) will "stick" to their positions in an image, even if the face moves. The goal is to have these fine feature positions be defined by coarse features. To do this, the paper overhauls the architecture of StyleGAN2, motivated by signal processing theory, and shows promising results. Specifically, the paper introduces 1x1 convolution kernels, more ideal low pass filtering for upsampling and downsampling, and filtering for nonlinearities such as ReLU activations. They also use a Fourier feature basis for the input vector, to restrict the frequencies learned by the network and create a continuous representation. We use several of these antialiasing techniques as the basis of our investigation in Chapter 6.

Chapter 3

Dataset

Although Li et al's Mannequin Challenge work [2] provides a dataset of videos, they do not provide corresponding depth data. As such, we generated depth data in order to train our model. We used COLMAP [4] to generate depth maps. COLMAP uses structure from motion to establish camera poses and a scene reconstruction estimate. COLMAP depth is incomplete; it only provides pixel depth when it can establish point correspondences. As such, it struggles in regions that do not contain texture, such as a blank wall or plain clothing. We empirically set a threshold mask to only train on regions where depth was defined by COLMAP. During training, we only computed loss for regions within the mask. After enough training, our model successfully generalized to fill the undefined regions in the training videos.

COLMAP is slow and not applicable for mobile videography use cases. It also assumes a static scene, which is why it pairs nicely with the Mannequin dataset but would not generalize well. Since COLMAP establishes scene reconstruction with camera poses, its depth maps are consistent. As such, we believe that this step does not inherently introduce inconsistency into our model.

Google's original dataset has over 170,000 video frames. Unfortunately, more than two-thirds of these videos are no longer available on YouTube. Our dataset consists of 49,915 training images and corresponding depths. As such, it is important to take any direct comparisons to the original model with this in mind. In our testing, the models we train from scratch do not meet the visual accuracy performance of Li et al.'s original weights in highly detailed areas, we assume due to our lack of

Figure 3.1: COLMAP estimate of an input frame from the Mannequin Challenge

training compute power and the size of our dataset. Our models also exhibit improved consistency over Li et al. even without any changes. This may be because our lack of data and training time does not allow the model enough time to learn a high frequency depth mapping function, causing outputs to look more similar. We generate a baseline model using Li et al.'s exact architecture and our dataset, so that we can accurately compare our results to a control and can be confident that differences in data size or training time are not contributing to improvements we observe in consistency.

For several of our experiments, we do not modify any trainable parameters. As such, we can load in Li et al.'s pre-trained model and fine-tune the network using various techniques. These fine-tuning tests allow us to take advantage of Li et al.'s pre-trained weights. These models typically exhibit more inconsistency (as expected, because the pre-trained weights display inconsistency), but we arrive at similar conclusions to training from scratch.

Chapter 4

Metrics

We believe that metrics do not tell the whole story, and most of our experiments relied both on quantitative measurement and qualitatively inspecting the videos to see if we agree. We found that the metrics we have listed below best align with our qualitative observations of our outputs. We outline techniques to measure both temporal consistency and accuracy (using COLMAP depth as ground truth).

4.1 Consistency

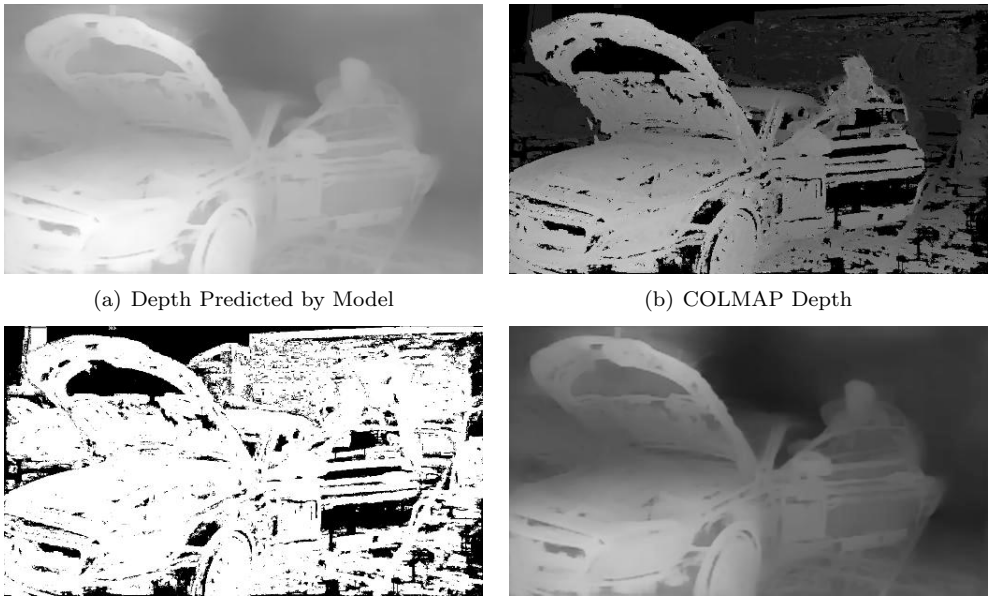
As stated when illustrating the problem, we choose to focus on videos with little to no motion of the camera or scene. As such, we can assume that the depth maps of consecutive frames should look more or less identical, especially if we blur these frames to remove small inconsistencies in the details. This assumption motivates our "Blurred L2" metric. First, we apply a Gaussian Blur to each frame. Next, we find the L2 distance between depths of neighboring frames. Intuitively, this metric represents a signal of how different two frame estimates are. Blurring removes high frequency details and subtle shifts between frames that would increase distance. In depth maps, we do not expect to see high frequency patterns, so this blur mainly helps to reduce the error seen between shifts in edges. As we are measuring relative depth, depths from successive frames could be different under some scale factor and bias. However, for a single model and near-identical inputs, we would expect the depth map to output under the same scale. As such, we do not account for a scaling factor between frames, and therefore penalize such a difference in this metric.

For a video X consisting of frames $x \in X$, we measure consistency $C(X)$:

$$C(X) = \frac{1}{|X|-1} \sum_{i=0}^{|X|-1} \sqrt{(G(x_i) - G(x_{i+1}))^2}$$

where $G(x)$ is a Gaussian Blur. $(G(x_i) - G(x_{i+1}))^2$ is a pixel-wise sum of squares.

We can read this metric in two ways. First, viewing the mean distance between frames can help us understand how much change in depth occurs, on average. Second, we can compute the variance in these observations, which can tell us how consistently the depth estimates are changing.



(a) Depth Predicted by Model

(b) COLMAP Depth

(c) COLMAP Depth Mask

(d) Scaled Model Depth

Figure 4.1: Accuracy Metric: Least Squares Regression

We would expect variance would be low, assuming camera motion isn't extremely erratic. Over the course of our experiments, we find that the mean distance better aligns with our qualitative consistency evaluations, so we mainly report this measure in this work.

We use this metric with videos with little motion (created by us) to make the problem simpler and our metric more reliable. We expect that results of our work generalize to dynamic scenes, but we have yet to develop a metric to measure this.

4.2 Accuracy

We use COLMAP depth as ground truth. COLMAP finds point correspondences between frames to estimate depth. Because of this, it struggles with untextured surfaces, which leads to undefined regions. As mentioned in Chapter 3, we create a mask of points with valid depth labels and only calculate accuracy for these points. Because our model outputs relative depth, different configurations may output depth at different scales. Therefore, we cannot naively compare depth maps from different configurations. We find that a least squares regression is the best way to align two relative depth maps from different models. Least squares computes a scale factor and bias to fit the model depth output to best match COLMAP depth as seen in Figure 4.1. Li et al. [2] only use a scale factor, but we find that using both a scale and bias better align our depth maps. We measure accuracy by computing the MSE between the scaled model depth and valid regions of the COLMAP depth. We use a test set of 10 videos from the Mannequin Dataset to measure accuracy.

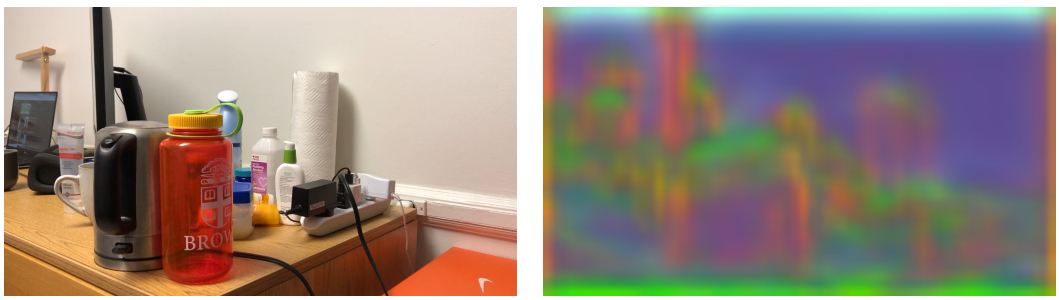
Chapter 5

Investigating the Latent Space

5.1 Visualizing the Latent Space

State of the art monocular depth estimation methods typically employ an hourglass-shaped autoencoder architecture. The encoder and decoder are typically both CNNs. Hourglass architectures work well for vision tasks because they increase the receptive field at each layer. Models with larger receptive fields perform well because they have a better semantic understanding of the scene, as each weight in subsequent layers is affected by many features from the previous layer.

This architecture forces the model to compress the input into a smaller internal representation, often known as the bottleneck. In order to better understand what information the model was retaining, we sought to visualize the latent space at the bottleneck of the hourglass. Our intuition was that this latent space would encode information about the scene. If we could determine that the latent encoding was already significantly different between similar input frames, we could narrow down the problem to the first half of the network. We used Google's pre-trained mannequin model for our visualization. The latent space of this network is high dimensional (256 x 32 x 64), which makes visualizing it a non-trivial task. Slicing along the first dimension to yield 32 x 64 greyscale images successfully showed us a representation of the scene, for some slices. For a more complete



(a) Input Image (b) Latent Visualization

Figure 5.1: Our Visualization of the Bottleneck Clearly Shows the Input Structure

picture, we used PCA. We projected the latent space onto its first three principal components.

We visualized these components as RGB values of an image, creating an image of dimensions (3 x 32 x 64), shown in Figure 5.1. Through this visualization, not only did we see the scene, we also noticed the flickering consistency issues that we hope to solve. Flickering was also observed quantitatively, by examining the L2 distance as a proportion of total possible distance between neighboring frames. We concluded that the latent encodings were significantly different when we could expect them to be similar. This further illustrates why the output of Li et al. shows inconsistency.

Additionally, we compared our latent visualizations to those outlined in Karras et al.'s [6] Alias-Free GAN. StyleGAN2's latent space well represents the face. StyleGAN3, however, looks significantly more abstract, although the faces are still visible. The authors hypothesize that their new model is encoding a coordinate system, which helps prevent texture sticking. The fact that our initial latent visualizations look more like StyleGAN2 could support our hypothesis that our model suffers from aliasing, although it is hard to know given that comparing our bottleneck to the Alias-FreeGAN architecture does not necessarily make sense due to the differing problem setting. Additionally, we do not know what method the authors used to visualize their latent space.

We chose to visualize the bottleneck specifically because it is the most compressed part of the network. It is also where the model switches from downsampling to upsampling, which will become important when we discuss antialiasing. Technically, all intermediate layers are "latent spaces." For the rest of this work, latent space refers specifically to the latent space at the bottleneck.

Visualizing complex models is difficult, so we were pleased with this result. Not only did this help diagnose the problem, we use this going forward to analyze our approaches.

5.2 Latent Space Regularization

Having shown that the latent encoding of consecutive frames was significantly different, we sought to create a model with a smoother encoder, with hopes that this would also produce smoother output depth. We compute a loss between latent spaces of neighboring frames, and add this to the original loss when training using a linear combination. We weight the latent loss with varying orders of magnitude as shown in Table 5.1. As this loss is computed at the bottleneck, it only back propagates to layers in the encoder portion of the network.

In initial experimentation, we overfit our models to single videos with little or no motion. Hypothetically, if frame b is slightly translated from frame a, one would expect that the latent encoding of frame b would be equally translated. However, translation is ill-defined in a high dimensional latent space. As such, we used frames that were nearly identical, arguing that their latent spaces should look nearly identical. In overfitting to a single video, we were able to improve consistency, but at the price of detail and accuracy. Qualitatively, more aggressive latent constraining weights led to blurrier depth maps and more consistency, whereas lower learning rates retained more high frequency information but were less consistent. We were able to verify this quantitatively. Given this information, we trained models on our large dataset using this constraint. We train from scratch

Configuration	Consistency (lower is better)	Accuracy (MSE, lower is better)
Li et al. [2]	2104	0.0189 +/- 0.0072
Ours, baseline (no latent constraint)	1162	0.0200 +/- 0.0082
Latent weight		
10^{-5}	836	0.0198 +/- 0.0074
10^{-3}	848	0.0216 +/- 0.0075
10^{-1}	722	0.0219 +/- 0.0080

Table 5.1: Training From Scratch with Latent Regularization

using the architecture of Li et al. and also fine tune their model weights.

5.2.1 Latent Regularization from scratch

We trained the architecture of Li et al. from scratch using our dataset. We reasoned that introducing a latent constraint at the start of training would be the best way to solve the problem, as the model weights of Li et al. were already plagued with inconsistency. We used a linear combination of the original loss and an L2 loss on the latent space. We use two baselines. In addition to the baseline of Li et al., we also train a model from scratch with no architecture or loss changes. This baseline is more useful because it represents changes in consistency and accuracy compared to the original work, which we believe are present due to a smaller training set and less training time. From Table 5.1, we can see that the weight of the latent loss correlated with improved consistency but decreased high frequency detail and accuracy. Intuitively, we believe that our additional loss was forcing our model to learn a smoother function. However, in doing so, it was over-generalizing, as it constrained neighboring frames to have the same latent encoding, when in reality they should not. We struggled to find a middle ground or escape this tradeoff. We saw similar results in a fine-tuning experiment.

Figure 5.2 shows the latent and supervision losses over the course of 10 epochs, for a latent weight of 10^{-1}. The latent loss decreases dramatically as we penalize it, corresponding to the immediate consistency benefits that we observe. In future work, it could be interesting to increase the latent weight throughout training time, to attempt to continuously improve consistency. Despite the latent loss becoming orders of magnitude smaller than the supervision loss, we find that throughout training time, consistency remains constant, whereas accuracy improves. The latent-constraint forces the weights to be learned in a different way initially, which remains through time. This leads us to believe that with enough training time, latent-constrained models could reach similar accuracy to baseline, while remaining more consistent.

5.2.2 Latent fine-tuning

Using a pre-trained model of Li et al., we performed a latent-constraint fine tuning step. Specifically, we added an L2 loss to constrain the latent space of neighboring frames, as in our previous

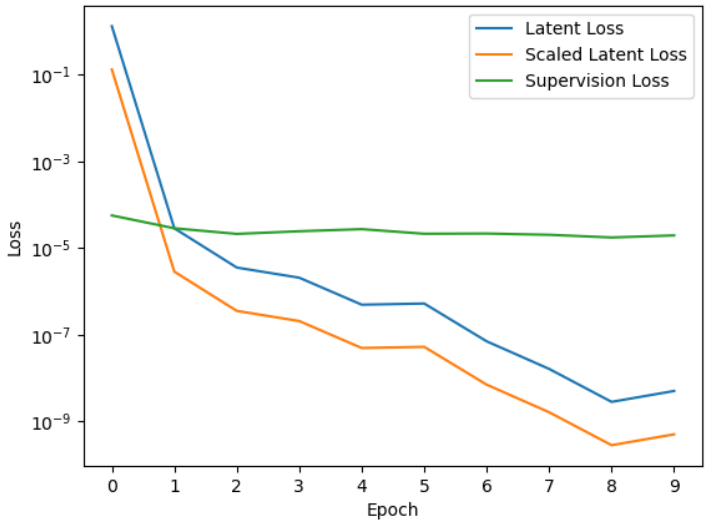


Figure 5.2: Supervision and Latent Losses over 10 Epochs

experiment. We train on our full dataset as described earlier. This dataset has considerable camera motion between frames. We believe this contributes to the blur we see in the output. Models with larger latent weight exhibit improved consistency. However, these improvements in consistency were strongly correlated with a drop in accuracy and loss of high frequency detail, mirroring initial tests overfitting to single videos and our from-scratch training.

5.2.3 L1 vs. L2

We compare two losses on consecutive latent spaces, L_1 and L_2:

$$L_1(a,b) = \sum_{i=0}^{n} |a_i - b_i|$$
$$L_2(a,b) = \sum_{i=0}^{n} (a_i - b_i)^2$$

Typically, L1 loss enforces sparser models, while L2 enforces smoother models [8]. We hypothesized that penalizing L1 loss on the latent space would lead to a sharper output image. Upon comparing the two losses in our fine-tuning experiment, L1 loss performed worse, both in accuracy and consistency. The L1 loss may be too harsh, enforcing sparse weights that prevent the model from sufficiently learning to represent the scene.

5.2.4 Results

We have shown that constraining the latent space leads to improved temporal consistency in the output. However, it also struggled to represent the same level of high frequency detail. This was true both when training from scratch, and when fine-tuning the weights of Li et al.

We also tried constraining the output and the latent space simultaneously, which yielded similar findings. Since the output depth is higher resolution than the latent space, small changes between

frames only increase the blurriness induced by forcing them to be the same. We chose to constrain the latent space because it enforces more consistency in the internal representation, and we also reason that it is more directly targeting the problem because we have proven the latent space is already inconsistent.

Tweaking the training process could certainly improve results. For example, finding a training set with videos with static camera poses could reduce the blurred output we see. We could also attempt to fine-tune the weight of the latent loss at different levels of the network, or perform an edge-preserving operation. Lastly, upon measuring consistency over training time, we found that consistency generally remained the same while accuracy improved. As such, training for longer could create models with strong consistency that show improved accuracy. While we believe that these directions could be promising, we believe it would improve our results only incrementally. Even in the aggressively latent-constrained versions with little high frequency detail, we see nonsensical flickering. This should not happen if the model is translationally equivariant. We hypothesize that this model architecture is doomed to be plagued by inconsistencies based on its construction. In the following sections, we outline these issues and propose overhauling several portions of the autoencoder structure to create a more translationally and rotationally equivariant model.

Chapter 6

Antialiasing

6.1 Background

It is common to sample from a continuous analog signal in order to digitize it. Every time we re-sample a signal, we are at risk for aliasing. Aliasing occurs when we undersample the original signal, which can lead to a representation that does not properly represent the input signal. By the Shannon-Nyquist Sampling theorem [9], for an analog signal with maximum frequency f_{max}, the sampling rate must be at minimum $2 * f_{max}$ in order to accurately reconstruct the signal.

Photography is an example of discretization of a continuous signal. When we measure light rays hitting a sensor to form a discrete set of pixel values, we are attempting to reconstruct the continuous scene using a sampling rate proportional to the number of pixels on the sensor. In video, we can observe spatial and temporal aliasing. Moiré patterns, are an example of spatial aliasing. A common example of temporal aliasing is the "wagon-wheel effect" [10], where spokes of a wheel or blades of a propeller appear to rotate at a different speed or direction to their true rotation. This can even seen with the naked eye, when looking at a car tire on the highway.

When we resize an image, we are at risk of aliasing. Neural networks frequently resize the input signal as it is passed through the network, making them vulnerable to aliasing issues. In particular, autoencoder networks like Li et al. [2] must downsample frequently to create the compressed bottleneck representation, and then upsample until the desired output size is reached. The strong compression and decompression of the signal makes these networks especially vulnerable to aliasing.

Based on the Alias-Free GAN work of Karras et al.[6], we believe that most neural networks used for video generation and understanding are prone to aliasing which causes inconsistency. This includes monocular depth estimation methods such as the architecture we use from [5]. Aliasing in depth estimation causes the network to learn high frequency signals that lead to a highly sensitive model, leading to inconsistencies in output. In the following sections, we use several techniques to combat these problems and subsequently improve temporal consistency.

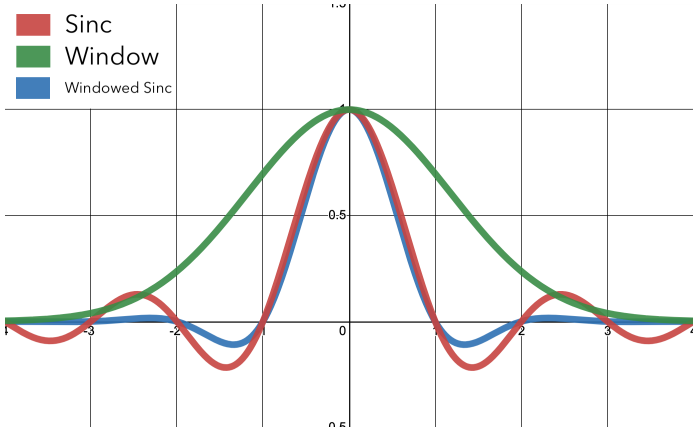


Figure 6.1: Sinc, Window, and Windowed Sinc functions, used for low-pass filtering

6.2 Antialias Resampling

6.2.1 The Sinc Filter

Naively up or downsampling an image will introduce aliasing. A naive method to downsample an image to be half its original dimensions would simply take every other pixel in each row and column to form a new image. This does not consider the potential frequencies in the output image. In order to prevent introducing new frequencies into the image, we must blur the output. A common technique in deep learning is to average neighboring pixels. This leads to a more satisfying result, however it may still introduce aliasing, as we do not explicitly remove high frequency signals.

A common image processing technique to prevent aliasing is a low-pass filter. Based on the Shannon-Nyquist theorem [9], a lower frequency signal is less likely to introduce aliasing when sampled (for a given sampling rate). A low-pass filter removes high frequencies from an image to allow a low sampling rate. In images, high frequencies are represented as areas of detail or texture. A low-pass filter in the image domain is a blur, which removes this detail.

In the Fourier domain, the ideal low pass filter is the box filter. This filter removes all frequencies above some cutoff threshold. However, a Fourier transform is an expensive operation. Given our kernel size, we choose to use the inverse Fourier transform of this filter, and convolve this with the input in the spatial domain. The spatial dual of the box filter is the sinc filter. The normalized sinc filter is: $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$. For some cutoff frequency f_c, the 1-dimensional ideal low pass filter [6] is:

$$h(x) = 2f_c \cdot \text{sinc}(2f_c \cdot x)$$

While theoretically the sinc function is the ideal low-pass filter in the spatial domain, it does not work in practice because it assumes the feature maps extend infinitely (which they do not). Convolving with this filter kernel in our model did improve consistency, however it also introduced unwanted ringing artifacts, which are expected when applying a truncation of this filter.

A windowed sinc function is a common approximation of the ideal low-pass filter. A window is some function $f(x)$ where $\lim_{x\to\infty} f(x) = \lim_{x\to-\infty} f(x) = 0$. Multiplying this function by the sinc filter creates a windowed sinc filter, which is an approximation of the ideal low pass filter without infinite spatial extent. This is shown in Figure 6.1; notice how the windowed sinc function quickly tends to 0. Common choices for a window function are the Blackman window and the Kaiser window. We choose a Kaiser window [11] (also known as a Kaiser-Bessel window) $w_K(x)$, based on Karras et al. [6].

Per [6], the 1-dimensional Kaiser window is:

$$w_K(x) = \begin{cases} I_0(\beta\sqrt{1-(2x/L)^2})/I_0(\beta), & \text{if } |x| \leq L/2 \\ 0, & \text{otherwise} \end{cases}$$

where L is the desired spatial extent, β is a hyperparameter controlling the shape of the window, and I_0 is "the zeroth-order modified Bessel function of the first kind" [6]. We then have the following approximation of the ideal low-pass filter.

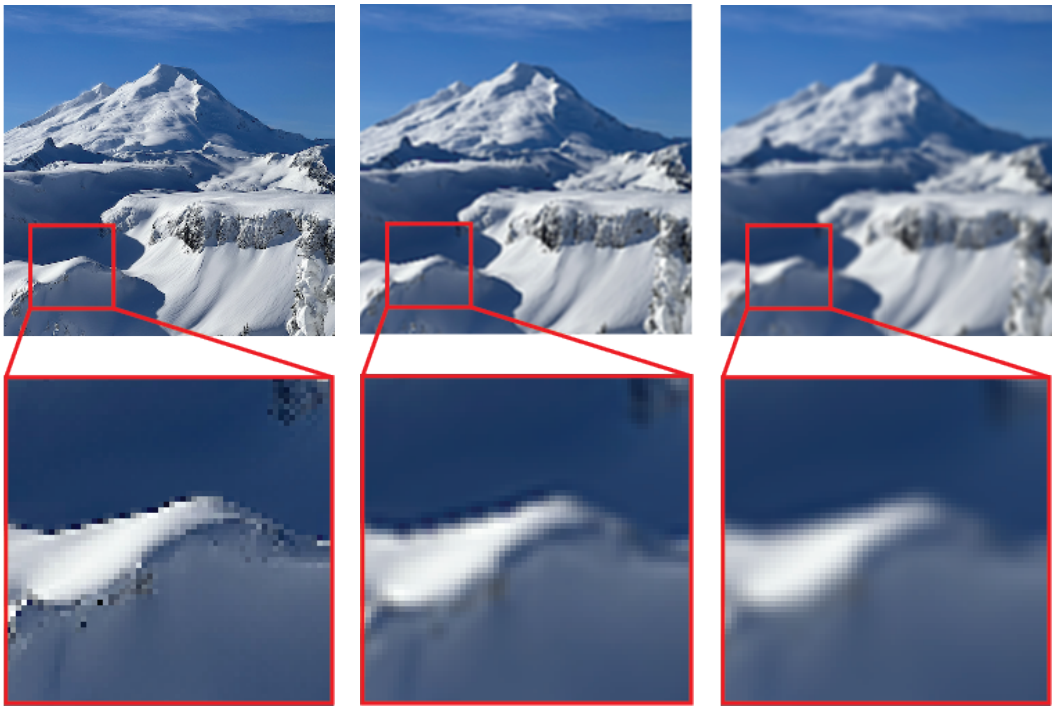
$$h_K(x) = w_K(x) \cdot h(x)$$

We must also discretize $h(x)$. Karras et al. uses the following discretization:

$$h_K[i] = h_K((i-(n-1)/2)/s)/s, \text{ for } i \in \{0, 1, ...n-1\}, \text{ creating n samples, } \tfrac{1}{s} \text{ units apart.}$$

They set the sampling rate s equal to the canvas width. The cutoff frequency f_c can be modified to employ critical sampling or non-critical sampling. Critical sampling sets f_c to the bandlimit, $\frac{s}{2}$. In other words, frequencies higher than $\frac{s}{2}$ are removed, to respect Shannon-Nyquist ($2f_{max} = s$). Karras et al. claim this is "ideal for many image processing applications as it strikes a good balance between antialiasing and the retention of high-frequency detail" [6]. However, since this work focuses on applying antialiasing techniques, we run most of our experiments using non-critical sampling with $f_c = \frac{s}{2} - f_h$, where f_h is the half-width of the kernel. Since f_h is a positive number, f_c will be lower than critical sampling, removing more high frequencies. Non-critical sampling exhibits a more aggressive blur; for our purposes, the more aggressive removal of high frequencies is motivated by our primary goal of preventing aliasing. Karras et al. also vary the number of feature maps inversely with f_c, but we do not. In practice, as evinced by our results in Tables 6.1 and 6.2, both critical and non-critical sampling schemes improve consistency. Although we only compare the two potential f_c values, one could imagine many architectures where the filter's frequency varies through the network, potentially allowing higher frequencies after the bottleneck, or only at the last few layers.

We compare our methods (convolving critical and non-critical sampling kernels with the input) to the original downsampling filter of [5], Average Pooling, in Figure 6.2. We can see that Average Pooling preserves more detail than the sinc filters; however, it also introduces aliasing such as the jagged edges shown in the cropped images. The same results can be observed in upsampling. To



(a) Average Pooling (b) Windowed Sinc, crit. sampling (c) Windowed Sinc, non-crit. sampling

Figure 6.2: Different Downsampling Methods

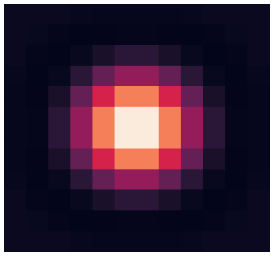


Figure 6.3: 2D Sinc Low-pass Kernel

Configuration			Consistency (lower is better)	Accuracy (MSE, lower is better)
Li et al. [2]			2104	0.0189 +/- 0.0072
Original architecture			1162	0.0200 +/- 0.0082
f_c	Sinc Upsample	Sinc Downsample		
$\frac{s}{2} - f_h$		✓	496	0.0228 +/- 0.0121
$\frac{s}{2} - f_h$	✓		1562	0.0201 +/- 0.0094
$\frac{s}{2} - f_h$	✓	✓	683	0.0226 +/- 0.0132
$\frac{s}{2}$	✓	✓	939	0.0227 +/- 0.0126

Table 6.1: Antialiased Resampling, from Scratch

upsample by a ratio of n, we create a new image, and use the original pixels as every nth pixel, and then apply the same Windowed Sinc filter to suppress aliasing.

In addition to translational equivariance, Karras et al. introduce the more aggressive rotational equivariance. To do so, they replace the sinc-based filter with a radially-symmetric jinc based filter, constructed with the Kaiser window as above. We use the radially-symmetric 2D filter as it is more aggressive in antialiasing. By the same logic as regarding critical sampling, both have benefits but for our purposes it makes sense to focus on the rotationally equivariant filter as it is more likely to improve consistency. Qualitatively, when applied to an image the two kernels produce similar results, with the rotationally equivariant kernel reducing detail only slightly more. Our radially symmetric kernel is visualized in Figure 6.3. It is important to note that this kernel does include small negative values, as can be visualized more clearly in Figure 6.1. For further rotational equivariance, Karras et al. introduce 1x1 convolution kernels, which we discuss later in Section 6.3.2.

6.2.2 Results

Tables 6.1 and 6.2 show our results. We train models replacing Bilinear Upsampling and Average Pooling Downsampling with our low-pass-filtered methods. We also train with improved upsampling and downsampling individually. Table 6.1 shows our results training from scratch using our dataset. Recall that $f_c = \frac{s}{2} - f_h$ corresponds to non-critical sampling, and $\frac{s}{2}$ to critical sampling. To attempt to take adventage of Li et al.'s pre-trained models, we also test fine-tuning the original weights in

Configuration			Consistency (lower is better)	Accuracy (MSE, lower is better)
Li et al. [2]			2104	0.0189 +/- 0.0072
Original architecture, fine-tune			1586	0.0163 +/- 0.0053
f_c	Sinc Upsample	Sinc Downsample		
$\frac{s}{2} - f_h$		✓	1508	0.0189 +/- 0.0068
$\frac{s}{2} - f_h$	✓		2389	0.0174 +/- 0.0064
$\frac{s}{2} - f_h$	✓	✓	1144	0.0191 +/- 0.0067

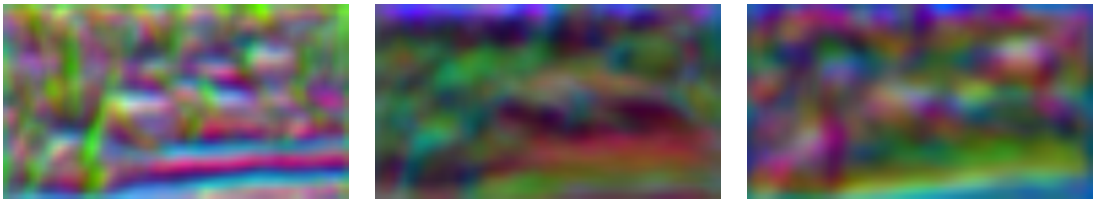
Table 6.2: Antialiased Resampling, Fine-tune

Table 6.2.

As a baseline, we train our model with no architecture changes. This baseline is more relevant than Li et al., because our models seems to always exhibit better consistency and worse accuracy than the original work, likely due to our dataset and training time. We can see that almost all antialias configurations outperform baseline consistency. Downsampling improves consistency in all trials. Upsampling, curiously, appears to fluctuate more. In fact, the configurations with only antialias upsampling actually perform worse than baseline. The fine-tune configuration with both antialias up and downsampling is the only indication that our upsampling method actually helps, not just downsampling. Since upsampling a signal increases the sampling rate, it is more difficult to introduce aliasing than when downsampling. We are unsure why our upsampling filter may actually worsen consistency, but it makes sense that it would do little to help remove high frequency signals. Throughout this work, we mainly focus on the encoder portion of the network, where we must compress signals, because this half of the network is more prone to aliasing. As such, it is a promising finding that downsampling improves consistency.

We sought to take advantage of Li et al.'s pre-trained models, as they exhibit good accuracy. Because the resampling layers that we modify are not trainable, we simply load in their weights and fine tune them using our modified architecture. As expected, fine-tune models are less consistent due to the original, inconsistent weights. It is important to note that while we can use the original weights, due to the changed sampling functions, the model must learn different weights in order to account for the change in signal frequency being passed through the model. Inference on the new architecture with the old weights initially yields blurry results, as the model of Li et al. does not need to learn to re-incorporate high frequency detail while decoding the latent space. Despite these caveats, our fine-tuned models yield similar conclusions to training from scratch. In both cases, we find that the consistency metrics we report match up well with our qualitative evaluation of the output videos.

Figure 6.4 shows latent space visualizations of several models, using the technique defined in Section 5.1. We can see both the latent-constrained and antialiased models exhibit lower-frequency activations in the latent space. Even though the antialiased model does not have an explicit latent



(a) Baseline (b) Latent Weight 1e-5 (c) Antialias Up and Downsampling, Non-critical sampling

Figure 6.4: Latent Space Visualizations

constraint, it makes sense that it learns lower frequencies, given that the low-pass filter used in downsampling removes high frequencies as the signal moves to the bottleneck. It follows that models with lower-frequency latent spaces have more consistent output, as there is less chance of high frequencies propagating aliasing through the network.

Unfortunately, our antialiased models are less accurate than our baselines. In fact, there is a strong correlation between improved consistency and worsened accuracy. Visually, many of these models appear to exhibit similar outputs to our baseline. However, we believe that the removal of high frequency signals prevents our models from learning the same level of detail as our baselines. We observe that these models appear more accurate than latent regularization models, although quantitatively they are worse. We believe that antialiased models do a better job of generalizing undefined regions of COLMAP, whereas latent regularization models overfit the dataset better, leading to their better performance according to our metrics.

In conclusion, while results were mixed for antialiased upsampling, it is clear that our downsampling method helps the model learn a lower frequency function, and thus a more consistent output. Although removing high frequencies leads to worsened accuracy, visually, we believe that this technique shows promise. We hope that increased training time and data could create a model that matches the accuracy of our baselines, while demonstrating our improvements in consistency.

6.3 Fourier Features

6.3.1 Background

Fourier features are a set of random sinusoids. These sinusoids can be used as a basis to transform a signal. Inspired by Karras et al., we use Fourier features as they allow us to explicitly control the frequency of the signal being passed through our network. We show that this further improves consistency. Karras et al. [6] use Fourier features as input to Alias-Free GAN. They mostly do this to introduce continuous translation and rotation when measuring equivariance, without the need for discretization. Fourier features are an inherently continuous representation. Continuous representations can be sampled without a risk of aliasing, which is an advantage for our purposes. This allows us to further gain translational and rotational equivariance.

Tancik et al. [12] show that it is difficult for a standard MLP to learn high frequencies in tasks

such as shape representation, texture synthesis, and, most relevant to our purposes, image fitting. This phenomenon of "extremely slow convergence to the high frequency components of the target function, to the point where standard MLPs are effectively unable to learn these components" is known as spectral bias. Tancik et al. show that this problem can be solved using Fourier features. Fourier features have been used in other settings to improve retention of high frequency detail, such as NeRF [13].

As defined in [12], a Fourier feature mapping $\gamma(\boldsymbol{v})$ from an input $\boldsymbol{v} \in [0,1)^d$ is:

$$\gamma(\boldsymbol{v}) = [a_1 \cos(2\pi \boldsymbol{b}_1^{\mathrm{T}} \boldsymbol{v}), a_1 \sin(2\pi \boldsymbol{b}_1^{\mathrm{T}} \boldsymbol{v}), ..., a_m \cos(2\pi \boldsymbol{b}_m^{\mathrm{T}} \boldsymbol{v}), a_m \sin(2\pi \boldsymbol{b}_m^{\mathrm{T}} \boldsymbol{v})]^{\mathrm{T}}$$

The frequency vectors $\boldsymbol{b}_j$ can be modified to control the frequencies learned by the MLP. Tancik et al. set $a_j = 1, \forall j$.

In high dimensional problems, densely sampling frequency vectors $\boldsymbol{b}_j$ is impractical, as the number of required samples scales exponentially with dimension. It is more practical to sample a set of random Fourier features from a parametric distribution. Tancik et al. show that this strategy matches performance of densely sampling Fourier basis functions. They find that a Gaussian distribution performs best; however, in practice, the standard deviation of the distribution is significantly more important than the distribution family. As seen in Figure 6.5, the standard deviation σ of the distribution dictates the frequencies learned by the network. In this example, the model was overfit to a single video of 20 frames, and the standard deviation clearly impacts the frequency spectrum of the output. Each $\boldsymbol{b}_j$ is sampled from $\mathcal{N}(0, \sigma^2)$. In a large training set, this difference may be less obvious when looking at a single image as the model has more time to generalize its output; however, at this scale of data the standard deviation plays a large role in consistency of the outputs.

Fourier features allow us to control the frequencies learned by the network. They are a powerful tool as they can help a network converge quickly. A kernel with a wide spectrum will achieve faster training of high frequency components; however, we must be careful not to reconstruct a signal too high in frequency, or we risk aliasing. As such, we must tune hyperparameters in order to find a model that estimates depth in high-enough frequency, but does not suffer from aliasing. One may reason that Fourier feature parameters could be optimized; however, Tancik et al. conclude that optimizing these parameters via gradient descent does not improve network performance [12].

In practice, we concatenate the RGB pixel values and the x, y position of each pixel, and pass this $\boldsymbol{v}$ as input to our Fourier feature mapping. Without this positional encoding, our model has no notion of spatial relationship and must only learn what depth a given color is most likely to be.

6.3.2 1x1 Convolution

Karras et al. replace all convolution kernels with 1x1 kernels to make the model rotationally equivariant. 1x1 kernels do not allow the typical spatial semantics learned by CNNs. However, the input to our Fourier feature mapping includes pixel coordinates, as part of the 5-dimensional vector. Due to the positional encodings being baked in to the Fourier features, they can still learn spatial information. 1x1 kernels allow for the full equivariance described by Karras et al. Unfortunately, in

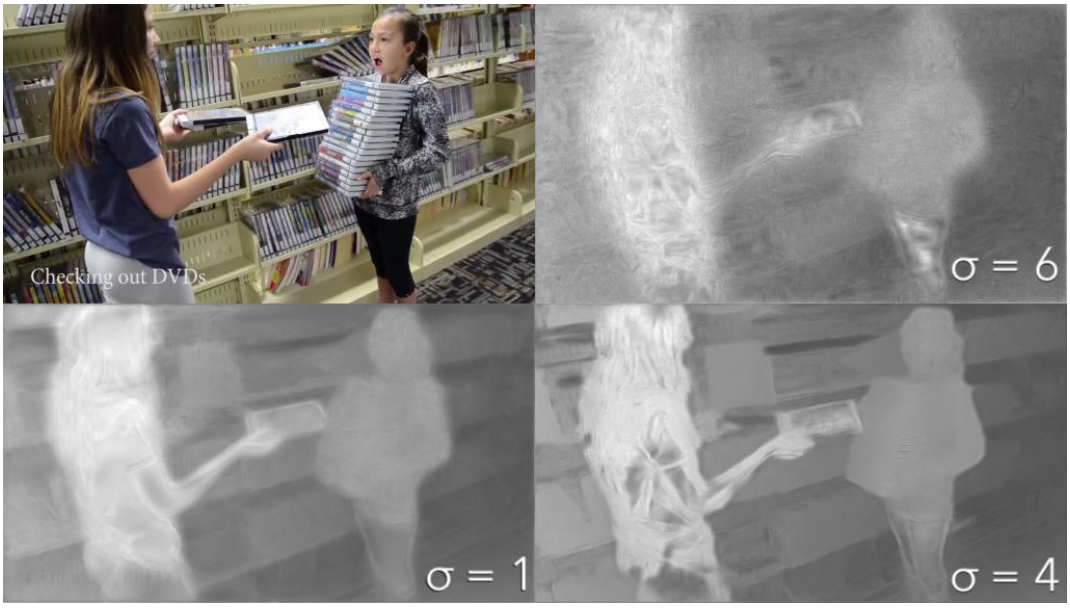


Figure 6.5: Overfitting to a single video, different σ values for Fourier bases

practice these models took signicantly longer to train were unable to converge to convincing outputs.

6.3.3 Results

As expected, higher σ values allow the model to converge quicker. σ of 4 or 6 present a fairly accurate model after only a couple epochs. However, these models introduce inconsistency. They appear to perform quite similarly to our baseline models, in that they converge quickly but are inconsistent. Tancik et al. [12] discuss this tradeoff. Unfortunately, models that were able to reduce spectral bias appeared to exhibit inconsistency, which we hypothesize is due to aliasing of high frequency signals. As rate of convergence is not our main goal, we did not investigate this further, and focused on low σ values.

Lower σ values correspond to more consistent output. This supports our hypothesis, that removing high frequency signals helps reduce aliasing and improve consistency. However, when combining this technique with antialiased resampling, results did not further improve. This may be because these two techniques seek to solve the same problem. Using small σ values is plagued by the same accuracy problems observed in resampling. As we decrease the frequencies learned by the model, the output lacks detail and accuracy worsens. Furthermore, using an increasingly small σ value (such as 0.001) does not converge to a convincing depth estimate and does not further improve consistency.

Because Fourier features use positional encoding, they allow us to use 1x1 convolution, removing all spatial dependency of the kernels and creating translational and rotational equivariance. We reasoned that 1x1 convolution would lead to the greatest equivariance and therefore the best consistency. Consistency results using 1x1 convolutions were mixed. However, these models were unable

Configuration					Consistency (lower is better)	Accuracy (MSE)
Li et al. [2]					2104	0.0189 +/- 0.0072
Original architecture					1162	0.0200 +/- 0.0082
1x1 Kernels	σ	f_c	Sinc Upsampling	Sinc Downsampling		
	0.001	$\frac{s}{2} - f_h$			714	0.0244 +/- 0.0110
	0.1	$\frac{s}{2} - f_h$			1042	0.0215 +/- 0.0095
	1	$\frac{s}{2} - f_h$			794	0.0221 +/- 0.0108
	2	$\frac{s}{2} - f_h$			1969	0.0250 +/- 0.0145
	4	$\frac{s}{2} - f_h$			1747	0.0185 +/- 0.0065
	0.1	$\frac{s}{2} - f_h$	✓	✓	690	0.0220 +/- 0.0099
	1	$\frac{s}{2} - f_h$	✓	✓	755	0.0254 +/- 0.0122
	1	$\frac{s}{2}$	✓	✓	718	0.0216 +/- 0.0081
	1	$\frac{s}{2} - f_h$		✓	782	0.0253 +/- 0.0126
	2	$\frac{s}{2} - f_h$	✓	✓	781	0.0236 +/- 0.0084
	4	$\frac{s}{2} - f_h$	✓	✓	1805	0.0209 +/- 0.0069
	6	$\frac{s}{2} - f_h$	✓	✓	1834	0.0284 +/- 0.0144
✓	2	$\frac{s}{2} - f_h$	✓	✓	912	0.0246 +/- 0.0094
✓	6	$\frac{s}{2} - f_h$	✓	✓	1176	0.0278 +/- 0.0124

Table 6.3: Fourier features performance

to converge to convincing depth estimates, so any consistency improvements are unimportant regardless. We believe this may be due to lack of training time. As such, we were unable to draw strong convolutions about the efficacy of 1x1 kernels.

To conclude, although Fourier features remove high frequency signals and improve consistency independently of improved resampling, combining the two does not help, and Fourier features improvements suffer from the same high frequency detail versus consistency tradeoffs.

Chapter 7

Conclusion

In this work, we propose several methods to improve temporal consistency in neural monocular depth estimation techniques. We provide a nonexhaustive visual representation of our contributions in Figure 7.1. First, we examine the bottleneck latent space and apply a regularization loss to attempt to improve the latent encodings. We show that the latent encodings between consecutive frames already show significant inconsistency, which is why we choose to focus on the first half of the network. With this technique we improve consistency, but lose high frequency detail in the process. This is because we constrain the model to learn the same latent encoding for neighboring frames, although they are not the same. Next, we hypothesize that aliasing due to high frequencies is the root cause of inconsistency, creating models that are overly sensitive to change in the input. We attack this problem by improving upon existing upsampling and downsampling methods, and adding a Fourier feature mapping to explicitly control signal frequency. These techniques improves consistency, as seen visually and quantified using our consistency metric. However, these techniques reduce high frequency signals throughout the model and create a blurrier output. Although the results look much better than latent-constrained models, we were still unable to escape this tradeoff or match the accuracy of our baselines. These findings can be useful not only for depth estimation models but for any single-frame deep learning models targeted at video that seek to establish temporal consistency.

In future work, we plan to further overhaul existing autoencoder architectures to make fully translationally equivariant models. We believe that antialiased models take longer to converge. As such, we hope that increased training data and training time could create models that improve consistency without the accuracy falloff that we see currently. We are also working on improved consistency metrics. We would like to create a metric that more accurately measures consistency for dynamic scenes, where neighboring frames may look significantly different. Flow could be used to improve upon our current metric. We are also investigating a depth reprojection method to align frames before constraining them, which could also be used as part of a consistency metric. This is detailed further in our Appendix.

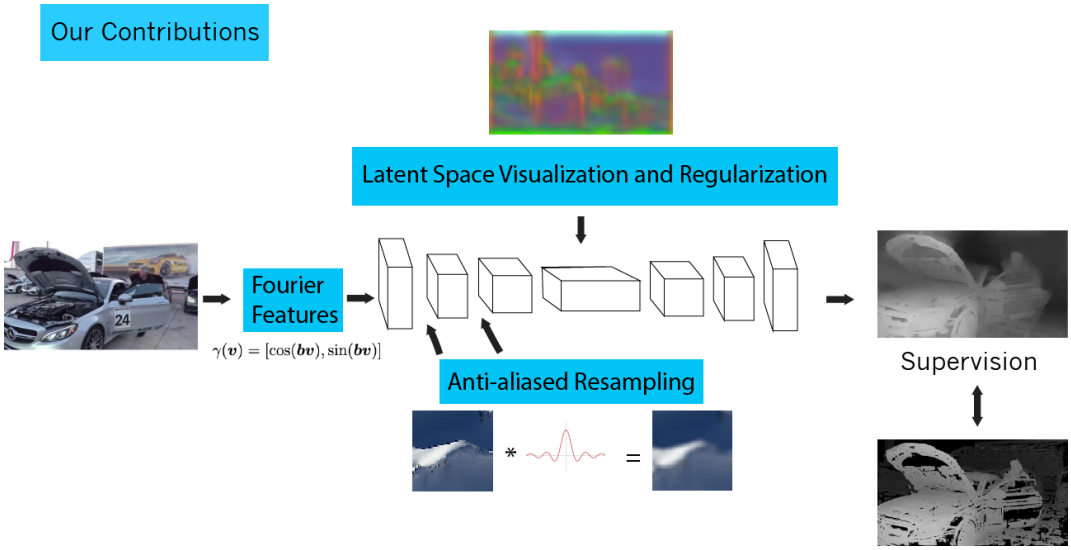


Figure 7.1: Summary of Contributions

Bibliography

[1] X. Luo, J. Huang, R. Szeliski, K. Matzen, and J. Kopf, "Consistent video depth estimation," vol. 39, no. 4, 2020.

[2] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman, "Learning the depths of moving people by watching frozen people," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *CoRR*, vol. abs/1604.03901, 2016.

[6] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Proc. NeurIPS*, 2021.

[7] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. CVPR*, 2020.

[8] R. C. Moore and J. DeNero, "L1 and l2 regularization for multiclass hinge loss models," in *Symposium on Machine Learning in Speech and Natural Language Processing*, 2011.

[9] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, pp. 10–21, jan 1949.

[10] Wikipedia contributors, "Wagon-wheel effect — Wikipedia, the free encyclopedia," 2021. [Online; accessed 11-April-2022].

[11] J. Kaiser and R. Schafer, "On the use of the i0-sinh window for spectrum analysis," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 1, pp. 105–107, 1980.

[12] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *NeurIPS*, 2020.

[13] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

Appendix A

Appendix

For our consistency metric, we apply a Gaussian Blur with $\sigma = 3$ before computing the difference between neighboring frames. We train all models from scratch for 20 epochs, except Fourier feature models with 1x1 convolution kernels. which we train for 30 epochs. We fine-tune Li et al.'s weights for 10 epochs. Training for 20 epochs takes between 3 and 7 days, depending on configuration, on one Nvidia Titan RTX. We use a learning rate 0.0004 for all models except Fourier features models, where we increase our learning rate to 0.001 for faster convergence. Our sinc filter kernel uses a width of 12. Karras et al. [6] found this to be the best tradeoff of an efficient approximation of the ideal low-pass filter. For COLMAP undefined regions, we empirically set a depth threshold to 0.3 for training.

A.1 Depth Reprojection Loss and Metric

One potential direction for future work is reprojection. COLMAP gives intrinsic (per video) and extrinisic (per frame) camera matrices. We use these to align depth maps from different frames, to compute a loss between them. This reprojection could prevent the blurring that we see if we naively constrain the outputs to be the same without reprojecting them into the same camera coordinate frame. This could also be used as a consistency evaluation metric. We have been unable to integrate this in our model thus far, but a visualization of reprojection can be seen in Figure A.1

(a) Depth from Frame A

(b) Depth from Frame B

(c) Depth from Frame B Reprojected into the Camera Space of Frame A

(d) Frame A blended with B

(e) Frame A blended with reprojected B

Figure A.1: Frame Reprojection, Frame B shown in red for clarity