Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Bachelor Thesis

Video Prediction using Local Phase Differences

Author:
Jan Nogga

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Joachim K. Anlauf

Advisor:
Hafez Farazi

Date: July 4, 2020

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Video prediction is commonly referred to as the task of forecasting future frames of a video sequence provided several past frames thereof. It remains a challenging domain as visual scenes evolve according to complex underlying dynamics, such as the egocentric motion of the camera or the distinct motility per individual object viewed. These are largely hidden from the observer and manifest as often highly non-linear transformations between consecutive video frames. Therefore, video prediction is of interest not only in terms of anticipating visual changes in the real world but has, above all, emerged as an unsupervised learning rule targeting the formation and dynamics of the observed environment. Consequently, state of the art approaches tend to either separate the image transformations from their causes, efficiently exploiting redundancies in changes of the video sequence but also limiting analysis to an explicit variety of global transformations, or rely on careful model design such that internal representations correspond to the hidden dynamics of the visual scene. To bridge this gap, we propose a flexible and content-independent local observation transform model inspired by the Frequency Domain Transformer Network architecture. We derive the relevant mathematical foundations and showcase results on synthetic as well as real data.

Acknowledgments

First and foremost, I would like to extend a huge *Thank You!* to my supervisor Hafez, for providing unrestricted access to his expertise in the domains of video prediction, and more generally, technical neural networks. I am fully convinced that if something has a gradient with respect to anything, Hafez knows how to train it and will concisely explain how (at any time of day). Throughout this work, I have benefited from this more often than I can count. Furthermore, I would like to thank my friend and colleague Peer and my father Andreas for proof-reading this thesis with a (I believe, you decide) keen eye. Also, my mother Astrid for her apparently infinite patience and endless support, along with my grandparents Anita, Hanne, Lothar and Gerd. Another shout-out is due to Simon and Dmytro, who were so thoughtful during their work on Baxter that I not once felt restricted in our shared access to robo9, which also deserves credit for tirelessly running its GPU without crashing once. Finally, I would like to thank Professor Behnke for encouraging me to actually apply math.

Contents

1. Introduction

1.1. Video Prediction

Video prediction can be described as the task of forecasting future frames of a video sequence provided several past frames thereof. Naturally, this involves developing an intimate appreciation of the underlying scene, which is generally composed of distinct entities interacting according to interdependent dynamics.

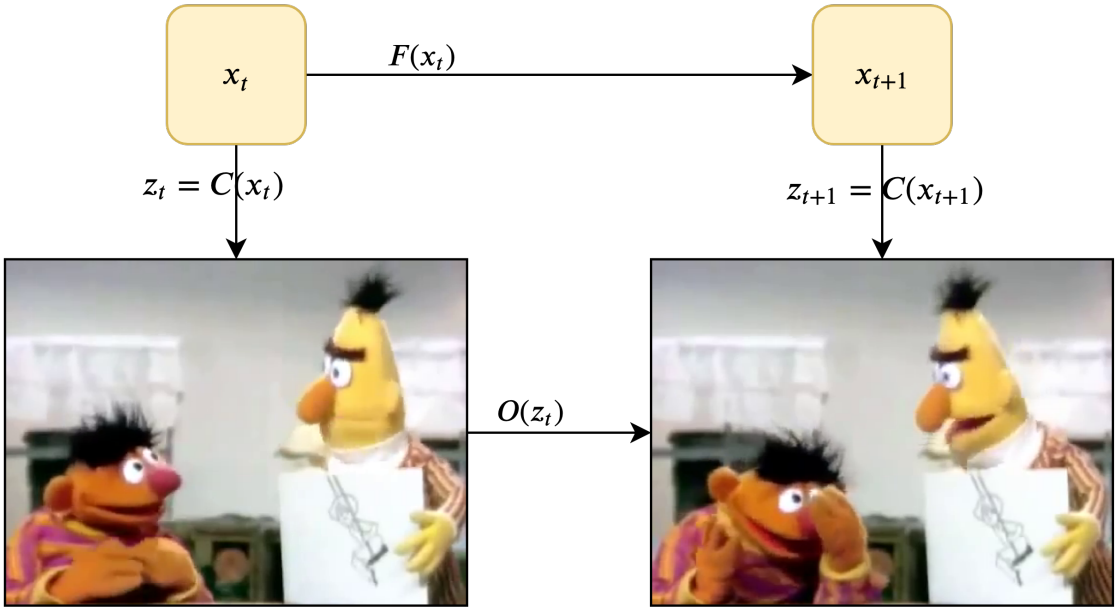

Figure 1.1: The dynamics of predicting video frames. Depicted scene from (Stone [1]).

This process is exemplified in Figure 1.1. Bert presents Ernie with an image of a little girl swinging. Intent on testing Ernie's ability to understand the pictured activity, he requests a description of what a drawing produced a short time later could depict. Ernie theorizes that, while the child is playing, her parents find they have run out of peanut butter. This is likely the girl's favorite spread, and thus Ernie concludes that her disappointment at dinner will be immeasurable. Only a second later, the viewer observes Bert's jaw dropped in disbelief as Ernie suffers a mental breakdown; the discrepancy between the child's current joy and her foreboding frustration has convinced him that all happiness is inevitably finite.

The wit of this sketch is that Ernie has turned the tables on Bert by producing an unpredictable outcome to his innocent prediction game. The transition between the jolly initial situation, referred to here as x_t, and the catastrophic conclusion in x_{t+1} is unexpected to muppet and human intelligence alike. The changes between the camera images z_t and z_{t+1} are minimal and, due to redundancies, can be encoded in the differences between only a small portion of the pixel values. The corresponding image-to-image mapping $\mathrm{O}(z_t)$, however, is but an ostensible manifestation of $F(x_t)$, the dynamics of the underlying scene. These are dominated by Ernie's eccentric character.

Of course, understanding absurd dynamics is not the intended use case of video prediction, but we can readily envision good predictions in other scenarios. For example, if we played Bert's game, we would notice that the girl's legs are tucked in and her upper body is pulled towards her knees. Thus, she is still on the upswing, and a consecutive frame would show the seat and the girl oriented by a slightly larger angle relative to the resting state of the swing, and accordingly, further above the ground. Taking into account some understanding of the use of swings, and provided the reference image, we can imagine exactly what the next drawing should look like just from a single shot. An artificial intelligent system with similar capabilities is the goal of video prediction, and since it is intuitive to judge its performance based on the faithfulness of the predicted frames, we can also interpret the process of predicting video sequence frames as a learning rule targeting comprehension of the underlying scene.

1.2. Related Work

This example illustrates that the output of a video prediction system can be evaluated based on the unlabeled frames, which are partially structured by their sequential arrangement. In this sense, learning to predict video is an example of semi-supervised learning, more specifically, of the self-supervised type (Engelen et al. [2]). On the other hand, it is clear that the ability to forecast future frames is limited by comprehension of the structure and dynamics of the underlying scene. If a trained predictive model is able to output accurate predictions, it must have formed some representations thereof (Lotter et al. [3]).

1.2.1. Video Ladder Networks

Video Ladder Networks (VLN) (Cricri et al. [4]) learn to model the hidden underlying state space by adding recurrent lateral connections to ladder networks. This is diagrammed in Figure 1.2. Specifically, lateral recurrent connections capture the

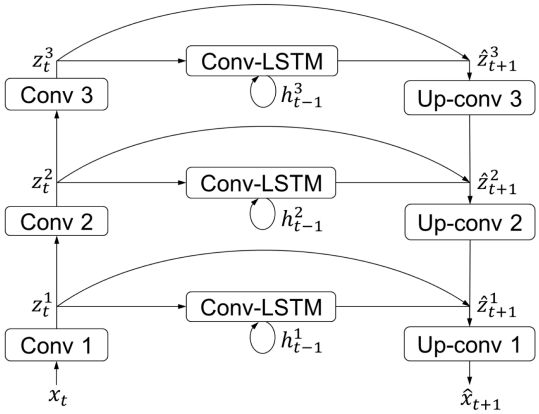

Figure 1.2: The video ladder network architecture (Cricri et al. [4]).

temporal dynamics of the scene at each level of abstraction, while lateral feedforward connections relieve upper layers from having to model low-level properties, most importantly spacial detail in the lowest layer. The authors demonstrate the ability of VLNs to predict subsequent frames on the Moving MNIST dataset (Srivastava et al. [5]). While the good performance shows that the model has an understanding of the hidden dynamics, these encoded representations cannot be accessed at all, thus the model suffers from a lack of interpretability.

1.2.2. Predictive Gating Pyramid

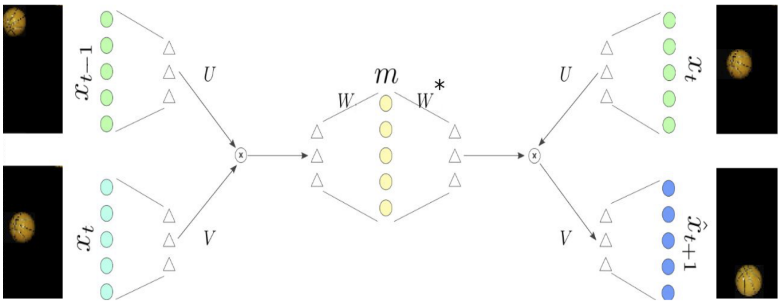

Figure 1.3: The relational autoencoder in the predictive gating pyramid (Michalski et al. [6]).

The predictive gating pyramid (PGP) (Michalski et al. [6]) uses a bi-linear model to encode the transformation between two observed frames. The hidden layer of mapping units in a gated autoencoder as shown in Figure 1.3 represents the transformation, which can directly be used to predict the next frame. This assumes that the transformation between x_{t-1} and x_t is linear, and remains constant toward

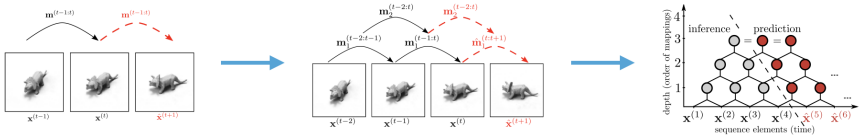

Figure 1.4: Representation of higher order derivatives in PGP (Michalski et al. [6]).

frame x_{t+1}. Higher order derivatives of the transformations are represented in an hierarchical manner, depicted in Figure 1.4. Here, the derivatives are regarded as transformations between two transformations observed from the lower layers. In stark contrast to VLNs, this deals with video prediction purely at the level of the image transformation, which themselves are restricted to linear transformations.

1.2.3. Local Transformation-Based Prediction

Another technique in which video prediction is reduced to the image transformation between consecutive frames is introduced by Amersfoort et al. [7]. Here, in a sliding window manner, small local cells are extracted from two consecutive frames. Between two local cells, an affine transformation is estimated, which is represented by six parameters and smoothed by multiple convolutional layers. The result can be applied to the latter of the two local cells, and finally all transformed local frames are used to construct the predicted next frame. Like in Reference (Michalski et al. [6]), the prediction model is small because it estimates the transformation towards the next frame instead of directly constructing the pixel values of the next frames. This also means that the structure of the scene cannot be addressed by these models. While the locally affine transformation model is more flexible than a global linear model, higher order derivatives of the image transformations cannot be represented directly.

1.2.4. Predictive Coding Networks

Both the lack of interpretability inherent to models with multiple layers of abstraction as well as the restrictive nature of assumptions made to justify transform-based prediction are adressed in Predictive Coding Networks (PredNet) (Lotter et al. [3]). In this network architecture, local predictions are compared to the input at that layer and only the deviation is passed upwards to higher layers. The connection between subsequent layers is sketched in Figure 1.5. At each layer l, a generative model of the convolutional layer A_l is learned by R_l. The prediction $\hat{A}_l$ is then compared to the true input from A_l and the resulting error E_l is passed into the next layer A_{l+1}, but also used to update R_l in the next time step. Addition-

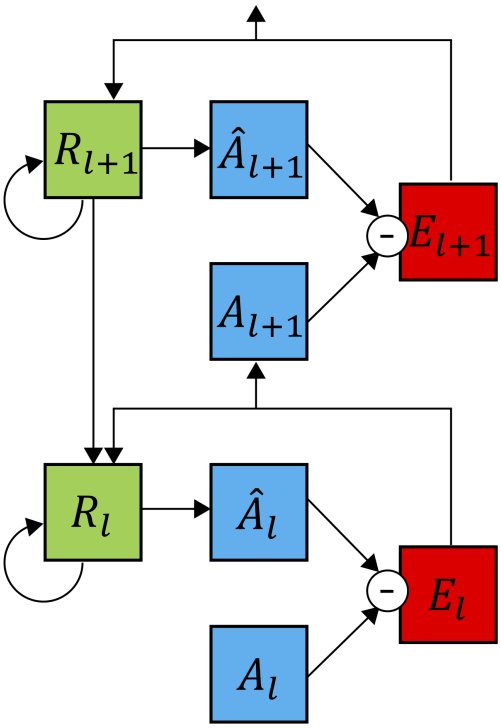

Figure 1.5: Two layers of the PredNet architecture (Lotter et al. [3]).

ally, R_l receives its previous output and an appropriately up-sampled version of the current output of R_{l+1}, one layer above. In the context of video prediction, the bottom-most layer would be presented the current frame as the input which is then compared to the video prediction $\hat{A}_0$. Higher layers then learn to represent latent parameters of the observed scene and objects present in the scene. For example, the authors train this model on a car-cam dataset, and demonstrate that the internal representations can be mapped to the approximate steering angle of the car using one fully connected layer. Additionally, the model trained on sequences of rendered 3D faces rotating with two degrees of freedom produces internal representations that enable a higher classification accuracy using less training samples for static faces than comparable ladder networks or autoencoders. In these examples, video prediction is used as a *proxy task* similar to the description by Jiang et al. [8], as the true purpose of training is learning representations of scene parameters useful to objectives such as steering angle estimation or orientation-invariant face classification.

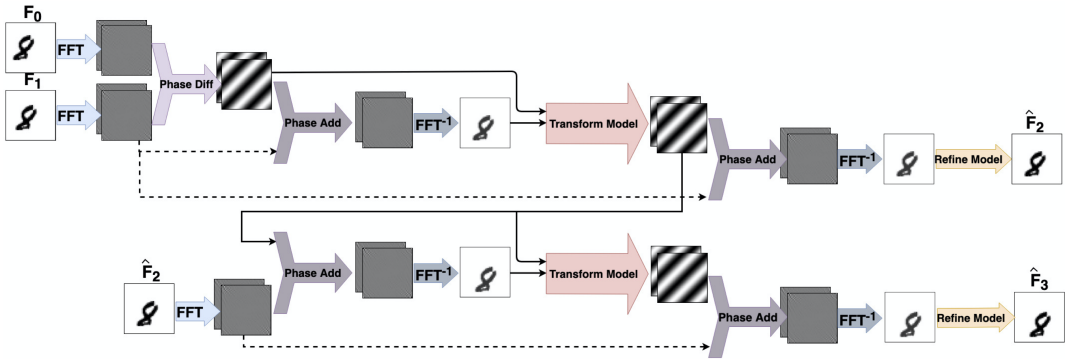

Figure 1.6: The FDTN architecture (Farazi et al. [9]).

1.2.5. Frequency Domain Transformer Networks

Frequency Domain Transformer Networks (FDTN) (Farazi et al. [9]) transform the video frames to frequency domain and estimate the transformation between the signals as the difference between their phases. The phase difference can be applied by an element-wise product to each bin of the frequency domain representation of the current frame. Subsequent inverse Fourier Transform of the result then yields the prediction of the next frame. Figure 1.6 shows the complete architecture, which is end-to-end trainable. The transform model adjusts the prediction for physics of the observed scene, for example digits bouncing off walls in Moving MNIST sequences. Finally, the refine model corrects slight image defects resulting from the prediction step. In a demonstration that this type of motion model can facilitate learning about scene structure in an interpretable manner, this concept has been augmented in recent work (Farazi et al. [10]) for segmenting moving objects from a static background. Purely self-supervised updates minimizing the prediction error are used for all components of the model.

1.3. Local Phase Difference

1.3.1. Motivation

The result of FDTN-based approaches are encouraging and illustrate the benefits of an explicit transform model at the core of an architecture learning to comprehend the underlying video scene. It allows light-weight models which are able to reason on the probable composition of the observed environment. However, the transform model using a global phase-add prediction is ultimately confined to describing the observed changes with one translation per identified layer.

To address this limitation, we take one step back from motion segmentation and re-examine the transformation model. Using a sliding window approach similar to the procedure in Reference (Amersfoort et al. [7]), we describe the transformation between video frames as local translational image movement measured by phase difference between local cells in the observed images.

This will remain differentiable, allowing its use at the base of models that can comprehend scene parameters. Unlike the affine tranformations estimated in the spatial domain used in (Amersfoort et al. [7]), this approach can be extended to represent higher order derivatives similar the results by Michalski et al. [6] as differences of differences described in Reference (Farazi et al. [9]) for the global case.

1.3.2. Contributions

We contribute a fully differentiable, GPU-accelerated pipeline implementing video prediction based on local phase differences. We also derive the mathematical foundations for local phase difference estimation and the synthesis operations required for reconstructing a global prediction from a set of local predictions. Finally, we list all caveats of this method and propose solutions for diminishing their negative effects.

2. Local Phase Differences

2.1. Phase Difference

For a sequence of frames $\{x_i\}$ of shape $U \times V$, assume that x_t results from a circular shift of x_{t-1}. Then

$$x_t[k,l] = x_{t-1}[(k+\Delta k) \mod U, (l+\Delta l) \mod V]. \tag{2.1}$$

For their Fourier partners $\{\mathcal{X}_i\}$, given by the discrete Fourier Transform (DFT)

$$\mathcal{X}_t[\omega_1,\omega_2] = \sum_{k=0}^{U-1}\sum_{l=0}^{V-1} x_t[k,l]e^{-j2\pi(\frac{\omega_1 k}{U}+\frac{\omega_2 l}{V})} \tag{2.2}$$

the shift theorem yields

$$\mathcal{X}_t[\omega_1,\omega_2] = \mathcal{X}_{t-1}[\omega_1,\omega_2]e^{2\pi j(\frac{\omega_1\Delta k}{U}+\frac{\omega_2\Delta l}{V})}. \tag{2.3}$$

Now we can define the element-wise phase difference as

$$\mathcal{PD}[\omega_1,\omega_2] := \frac{\mathcal{X}_t\overline{\mathcal{X}_{t-1}}}{|\mathcal{X}_t\overline{\mathcal{X}_{t-1}}|} = \frac{\mathcal{X}_{t-1}\overline{\mathcal{X}_{t-1}}e^{2\pi j[\frac{\omega_1\Delta k}{U}+\frac{\omega_2\Delta l}{V}]}}{|\mathcal{X}_{t-1}\overline{\mathcal{X}_{t-1}}e^{2\pi j[\frac{\omega_1\Delta k}{U}+\frac{\omega_\Delta l}{V}]}|} = e^{2\pi j[\frac{\omega_1\Delta k}{U}+\frac{\omega_2\Delta l}{V}]} \tag{2.4}$$

where $\mathcal{X}_i$ is implicitly indexed by $[\omega_1,\omega_2]$. This is omitted for convenience of notation.

The inverse DFT of the phase difference yields the cross correlation matrix $\mathrm{pd}[u,v]$. For perfect circular shifts, this turns out to be

$$\mathrm{pd}[k,l] = \delta[k+\Delta k, l+\Delta l] \tag{2.5}$$

a Kronecker delta function, its peak corresponding to the shift between x_{t-1} and x_t. Analysis of circular shifts by this method is also referred to as *Phase-Only Correlation* (Takita et al. [11]). Variants that use coordinate transforms to extract scale or rotation in a phase-based manner have also been described, e.g. by Reddy et al. [12].

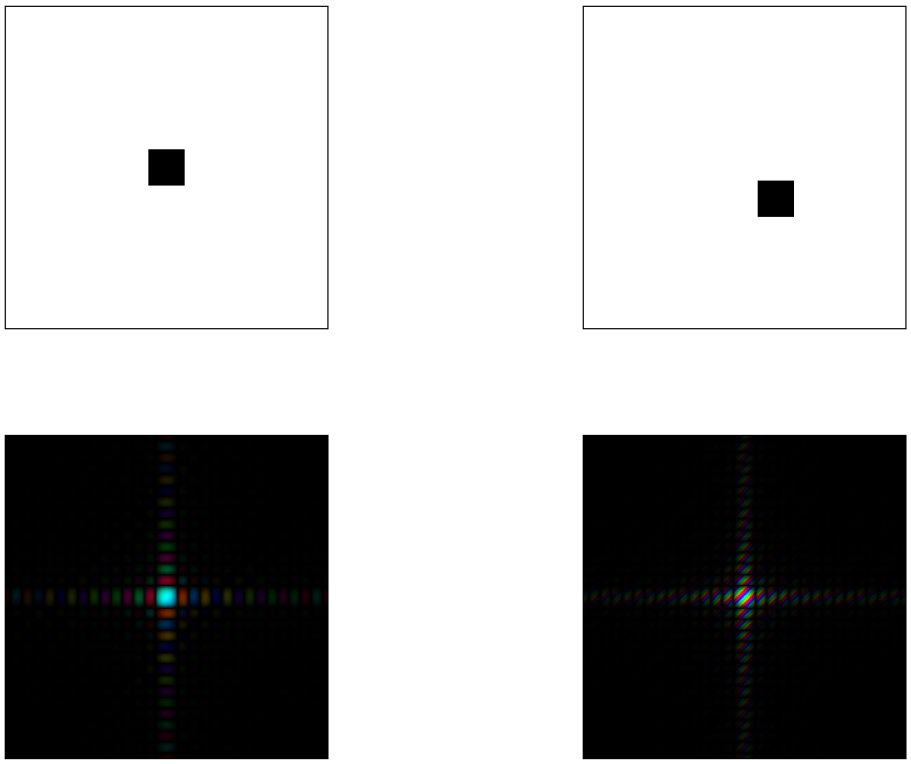

Figure 2.1: An example image x_0 (*top-left*), image with translated contents x_1 (*top-right*), along with their corresponding Fourier partners $\mathcal{X}_0$ (*bottom-left*) and $\mathcal{X}_1$ (*bottom-right*).

Note that in practice, the fast Fourier Transform (FFT) (Cooley et al. [13]) is used to compute the DFT. In the scope of this thesis, both terms are considered interchangeable.

A basic example is presented in Figure 2.1. An image x_t portraying a square at the center is shifted towards the bottom-right to produce x_{t+1}. A visualization of the Fourier-transformed frames $\mathcal{X}_t$, $\mathcal{X}_{t+1}$ shows that the power spectra encoded by the intensity $I \propto \log(|\mathcal{X}_i|)$ are equivalent for both Fourier partners. The phase is mapped to an angle via $\mathcal{X}_i[\omega_1, \omega_2] = a + bj \rightarrow \text{atan2}(b, a)$ and visualized by an associated hue. This illustrates the effect predicted by equation (2.3). Next, Figure 2.2 illustrates the real part of the resulting phase difference $\mathcal{PD}$ alongside pd. As expected, the latter corresponds to an impulse located at the index of the shift, highlighted by a red circle. It should be stressed that the white static background of the square renders the transformation indistinguishable from a circular shift. In more realistic cases, we expect the results to be noisy.

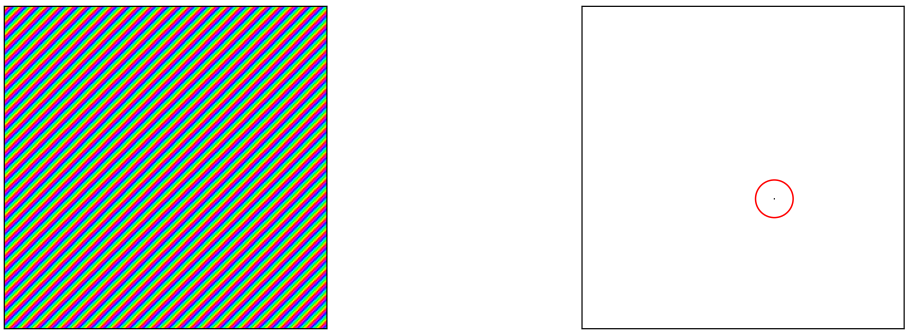

Figure 2.2: The real part of the phase difference $\mathcal{PD}$ extracted for a circular shift (*left*) and the corresponding cross-correlation pd (*right*). Here, pd is a perfect impulse, its peak highlighted by the red circle.

2.2. Local Analysis

2.2.1. Valid Extraction of Local Cells

We can now move to apply the same analysis to small overlapping cells covering an image on a regular grid. We begin by formalizing the extraction of these local cells, using notation which is borrowed from (Sharpe, Li [14, 15]). While it might seem pedantic here, the discussion in the subsequent chapter will benefit from this approach.

Consider an image x covered by overlapping, square cells of size $N \times N$. N is called the *window size*. In general, the window dimensions do not have to be square, but we see no obvious advantage in introducing this imbalance. The overlap between the cells along the coordinate axes is $N - H$, we call $H \leq N$ the *hopsize*. In the context of convolutional neural networks, this would correspond exactly to the *stride* of a kernel. H is also the distance between the centers of each cell to the adjacent cells centers along the coordinate axes. Restricting the hopsize in this manner means that each pixel in x is contained in at least one cell. Additionally, we impose the constraint that the first local cell should always be extracted around the top left pixel, which corresponds to the index $(0, 0)$ in the image coordinate system. The final cell that is extracted must be centered on the right-most pixel in the bottom row of the image. For this extraction to make sense, the image is padded with $\lfloor \frac{N}{2} \rfloor$ constant values. Since the extraction grid is regular, this constrains the hop sizes producing a valid extraction to divisors of $U - 1$, and $V - 1$, as the images are represented by zero-indexed arrays. This ensures that the final rows and columns of pixels are never lost, which would

prevent reconstruction. In practice, we often crop the image data to $U = 2^n + 1$, $V = 2^m + 1$ to ensure that many options for H exist over several magnitudes of sparsity. Overall, $L_y \coloneqq \frac{U-1}{H} + 1$ and $L_x \coloneqq \frac{V-1}{H} + 1$ cells are extracted along each axis, with $L \coloneqq L_y L_x$ representing the total number of local cells.

Formally, we can describe the contents of a given cell by shifting the coordinate system of x such that the new origin is aligned with the cell and then regarding only N elements along each axis

$$x_{u,v} = \{x[n + u \cdot H, m + v \cdot H] | n, m \in \{0 \dots N-1\}\}. \tag{2.6}$$

For more convenient notation, we could also extract $x_{u,v}$ by defining a window w of size $N \times N$, shifting it to cover exactly $x_{u,v}$, and then multiplying it with x

$$x_{u,v}[n, m] = x[n, m] \cdot w[n - u \cdot H, m - v \cdot H]. \tag{2.7}$$

For absolute clarity, w is defined on

$$\{(n, m) | n, m \in \{0 \dots N-1\}\}. \tag{2.8}$$

Therefore, to extend x, $x_{u,v}$ and w to $\mathbb{Z} \times \mathbb{Z}$ to allow for products like above, we consider the image, the cells and the windows infinitely zero-padded. In practice, we of course only work with the supports of terms at hand.

2.2.2. Local Fourier Transform

Following the cell extraction step, the corresponding local DFTs are given by

$$\mathcal{X}_{u,v}[\omega_1, \omega_2] = \sum_{n=uH}^{uH+N-1} \sum_{m=vH}^{vH+N-1} x_{u,v}[n, m] e^{-j\frac{2\pi}{N}(\omega_1(n-uH)+\omega_2(m-vH))} \tag{2.9}$$

$$= \sum_{n=uH}^{uH+N-1} \sum_{m=vH}^{vH+N-1} x[n, m] w[n - uH, m - vH] e^{-j\frac{2\pi}{N}(\omega_1(n-uH)+\omega_2(m-vH))} \tag{2.10}$$

$$= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x[n + uH, m + vH] w[n, m] e^{-j\frac{2\pi}{N}(\omega_1 n+\omega_2 m)} \tag{2.11}$$

In tandem with the previous step, this defines a two-dimensional version of the *Short-Time Fourier Transform* (STFT) (Bracewell [16]). Since it is a self-evident extension of the STFT, renaming would not be warranted, but the indices of individual images are spatial rather than temporal, so this is not a fitting denomination. In the following, we refer to this process as a *Local Fourier Transform*

(LFT). It should be noted that a texture feature of that name exists (Feng Zhou et al. [17]). This work is unrelated, and the risk of confusion is low.

Given $\mathcal{X}_{t-1,u,v}$ and $\mathcal{X}_{t,u,v}$, the LFTs of two consecutive frames x_{t-1} and x_t, the local phase difference is then defined element-wise as

$$\mathcal{PD}_{u,v} \coloneqq \frac{\mathcal{X}_{t,u,v}\overline{\mathcal{X}_{t-1,u,v}}}{|\mathcal{X}_{t,u,v}\overline{\mathcal{X}_{t-1,u,v}}|}. \tag{2.12}$$

2.2.3. Caveats

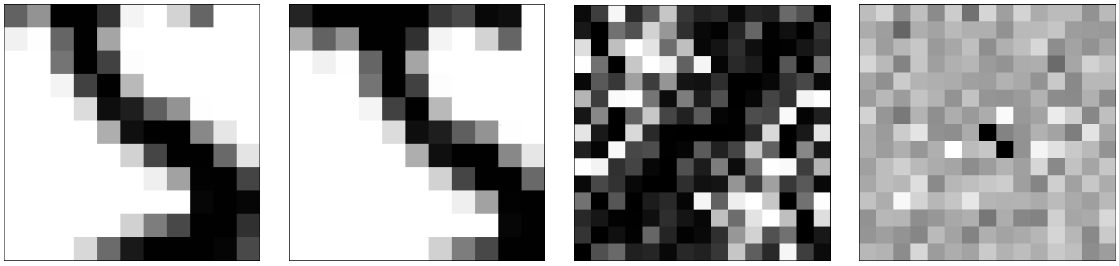

Figure 2.3: *From left to right*: a local view on frame x_0 with rectangle windowing, the view at the same spatial index on x_1, the noisy phase difference extracted between them and the resulting ambiguous cross-correlation.

Globally as well as locally, the extraction of $\mathcal{PD}_{u,v}$ assumes a transformation in the form of a circular shift between consecutive views. While the description of image transforms as locally linear shifts is not restrictive, the assumption of circular cell boundaries is, as the DFT also assumes input of a periodic signal. Figure 2.3 shows the consequences of relaxing this assumption without further consideration. Noise in the phase difference produces a local cross correlation that is not a Delta function, but instead quite distinctly ambiguous.

We can attenuate this shortcoming by weighting each local cell with a window function that smoothly tapers the intensity values towards the edge of the cell. In this section, we utilize a family of confined Gaussian windows described in Reference (Starosielec et al. [18]). To adapt these for use on two-dimensional cells, we use a radial mapping that places the peak of the window function values in the center of the cell and computes the other values based on the distance from the center. This family of windows is depicted in Figure 2.4. In Figure 2.5, the effect of applying the window to the problematic cell is demonstrated. The phase difference is less noisy and, in turn, the ambiguity in the cross correlation matrix is largely resolved.

This way, shift estimation is quite robust, as long as the shifting information content remains within the local cell. This is showcased in Figure 2.7. In Figure

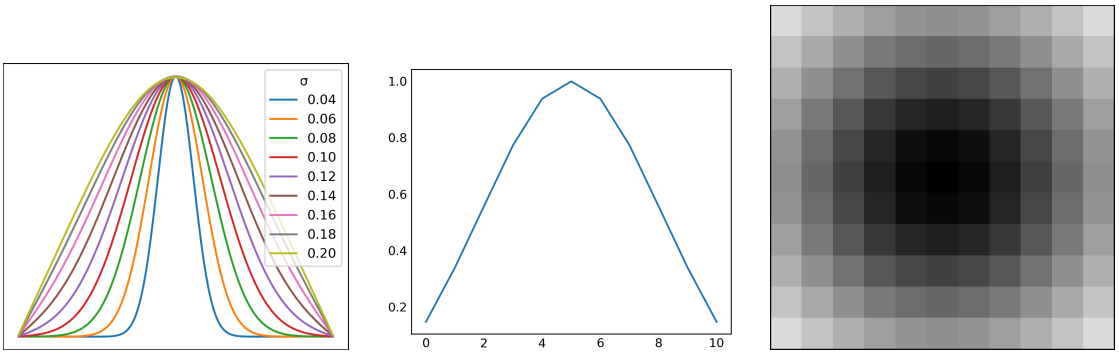

Figure 2.4: *From left to right*: an overview of the family of confined Gaussian windows, the cross-section of a window for $\sigma = 0.17$ and $N = 11$, and its 2D plot.

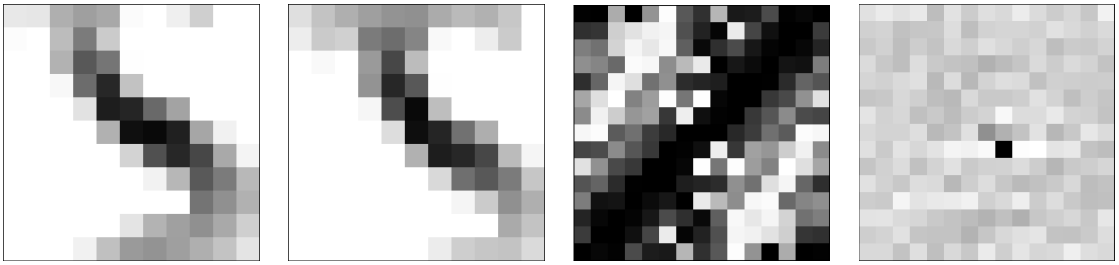

Figure 2.5: *From left to right*: a local view on frame x_0 tapered by a confined Gaussian window, the tapered view at the same spatial index on x_1, the phase difference extracted between them and the resulting cross-correlation.

2.7, a large structure is shifted to the outside of the cell, which causes a misinterpretation of the local shift. Nevertheless the energy in the cross correlation matrix at the index of the true shift is also elevated.

2.2.4. Vector Field Interpretation

As indicated by the previous examples, a qualitative evaluation of the results during the local shift estimation entails examination of the cross-correlation in the spatial domain. There, we can verify that it is indeed peaked around the index of the true shift. If we extracted the arguments corresponding to the peak, this would yield the relative shift at the pixel location in the original image that the corresponding cell was centered on. Assembled over the whole image, a vector field describing the image translation is accrued. After calculating $\mathcal{PD}_{u,v}$ and transforming these to the cross-correlation matrices $\mathrm{pd}_{u,v}$, local relative shifts are given by

$$(\Delta y, \Delta x)_{u,v} = \underset{(y,x)\in Ind}{\operatorname{argmax}} \mathrm{pd}[y, x]_{u,v}. \tag{2.13}$$

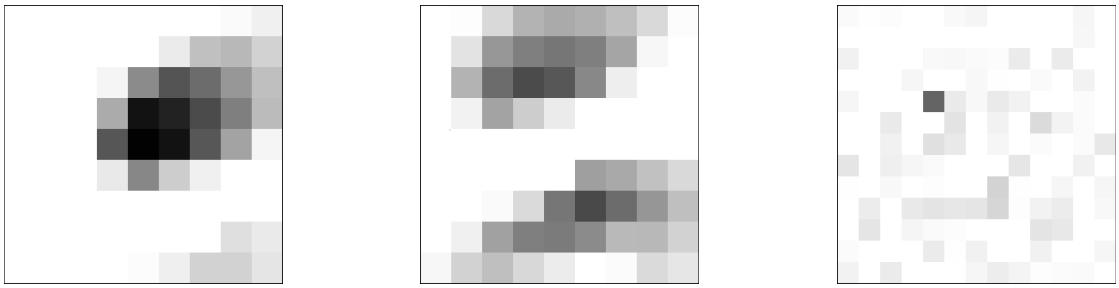

Figure 2.6: *From left to right*: a local view on frame x_0, the same view at frame x_1, and the corresponding cross-correlation pd, showcasing robust extraction of the local shift.

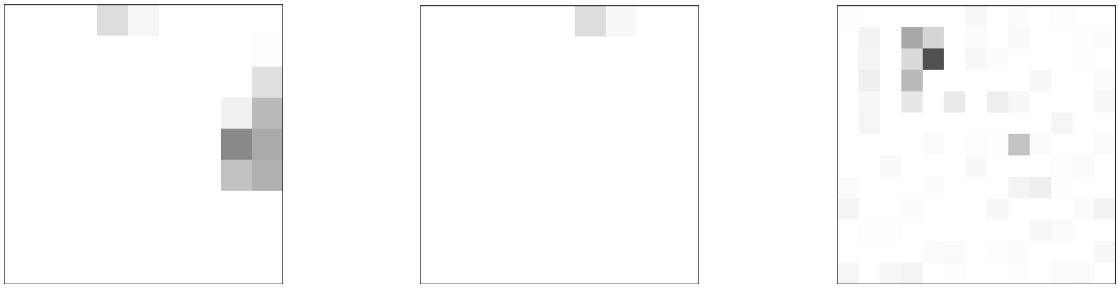

Figure 2.7: *From left to right*: a local view on frame x_0, the same view at frame x_1, and the corresponding cross-correlation pd, showcasing a misjudgment of local shift resulting from cell contents leaving the local view.

In Figure 2.8, we present this type of visualization. 3 digits are shifted, rotated and scaled, respectively. In the manner described above, the descriptive vector field is extracted and superimposed on the seed image. Vectors describing zero shifts are omitted. The drawback in this naïve approach is clearly the loss of sub-pixel accuracy inherent to the phase-based encoding of the shifts. To preserve this property, we use a soft argmax (Sutton et al., Goodfellow et al. [19, 20]) instead

$$\text{softargmax}_\tau(\text{pd}_{u,v}) \coloneqq \sum_{(y,x)\in Ind} (y,x) \frac{e^{\frac{\text{pd}_{u,v}[y,x]}{\tau}}}{\sum\limits_{(\tilde{y},\tilde{x})} e^{\frac{\text{pd}_{u,v}[\tilde{y},\tilde{x}]}{\tau}}}. \tag{2.14}$$

The parameter τ is called the computational temperature and steers the bias of the result towards the argmax of $\text{pd}_{u,v}$. For $\tau \to 0$, the result is the argmax, and for $\tau \to \infty$, the result is pulled towards the zero shift index pair $(0,0)$ corresponding to the center of the cell. To produce valid relative shifts, the index set Ind of $\text{pd}_{u,v}$ is offset by $\lfloor \frac{N}{2} \rfloor$. The added benefit of such an extraction is that it is fully differentiable, also with respect to τ. These relative shifts can thus be used within any training procedure that uses backpropagation, and τ can be automatically

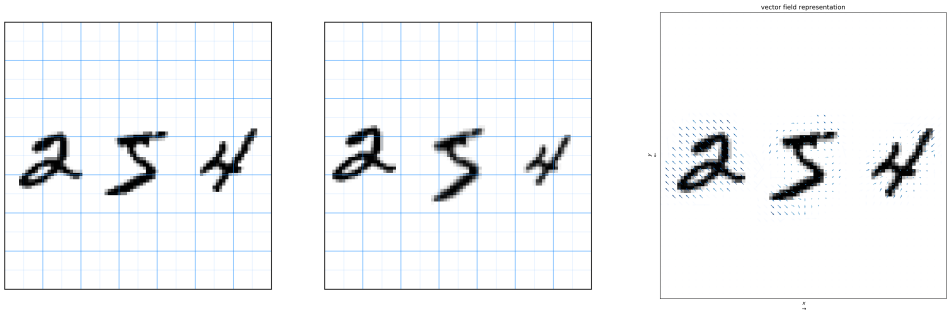

Figure 2.8: An image x_0 and its transformed version x_{t+1} (*left and center*). The blue grid is superimposed to clarify the transformation. A vector field visualization of the local shift extraction (*right*).

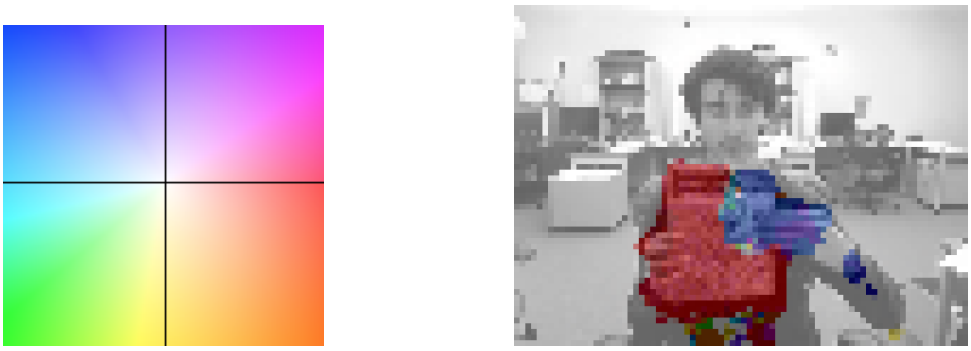

Figure 2.9: The optical flow color map proposed by Baker et al. [21] and application of this encoding to estimated local shifts.

adjusted as a further learnable parameter.

We experiment with a color coding for the relative shift as suggested in Reference (Baker et al. [21]). Figure 2.9 describes the corresponding color map and the result of sparsely applying the local relative shift estimation on video data of a person moving different objects in front of a camera. Another example is presented in Figure 2.10, where a skier is filmed while jumping off a ramp. Both sequences are taken from object tracking benchmark data provided by Wu et al. [22]. The relative shifts not only show the person flying forwards, but also display the projection of the egocentric motion of the camera following the skier onto the textured background of the scene.

Nevertheless, it would not be justified to equate the shifts encoded in the phase difference to faithful representations of optical flow. This is partially relaxed in work by Reyes et al. [23]. Here, an LFT variant without tapering and utilizing the argmax generates candidate vectors for further iterative optical flow refinement.

(a) ski flow

Figure 2.10: *From left to right*: image of person skiing at time t, image of the same scene at time $t + 1$, and the color coded local shifts between the frames.

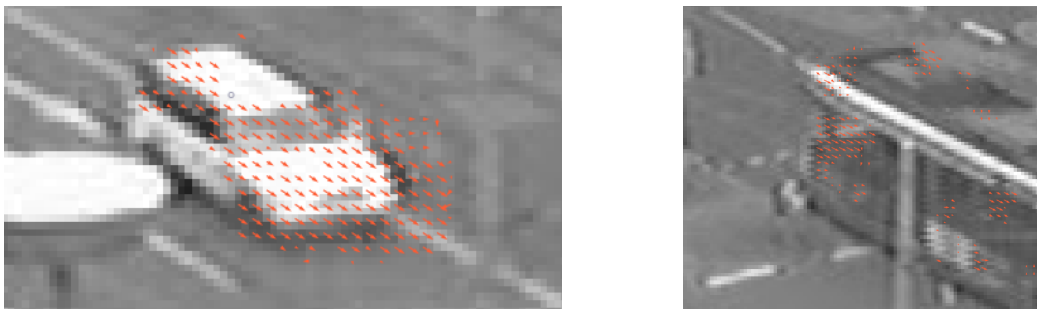

Figure 2.11: Two examples of local shift misjudgments in monochrome image regions for small window sizes.

However, cases as pictured in Figure 2.11 highlight that $pd_{u,v}$ does not encode correct pixel velocity when there are monochrome regions wider than N. The video sequence is provided by Fleischer et al. [24]. In phase-based methods directly targeting optical flow (Fleet et al. [25]) and specifically in Reference (Fleet et al. [26]), a more involved procedure is employed to evaluate space-time surfaces of constant phase. The pixel velocity is then calculated based on these contours. This requires extensive numerical optimization. However, we are concerned with image prediction, and monochrome image regions only need to be transformed at their edges. Furthermore, exploring the avenue of predictions via the DFTs of cross correlation matrices constructed according to pixel velocities gained from optical flow algorithms is not within the scope of this thesis, even though it might yield improved shift extraction at the price of heavy computational effort. In conclusion to this section, we stress that the relative shifts are a useful tool to visually evaluate intermediate results of the prediction, but should not be mistaken for optical flow.

2.3. Validity of Local Phase

In this chapter so far, we have introduced a method to extract the local phase difference between two frames. This was motivated by the necessity of a flexible

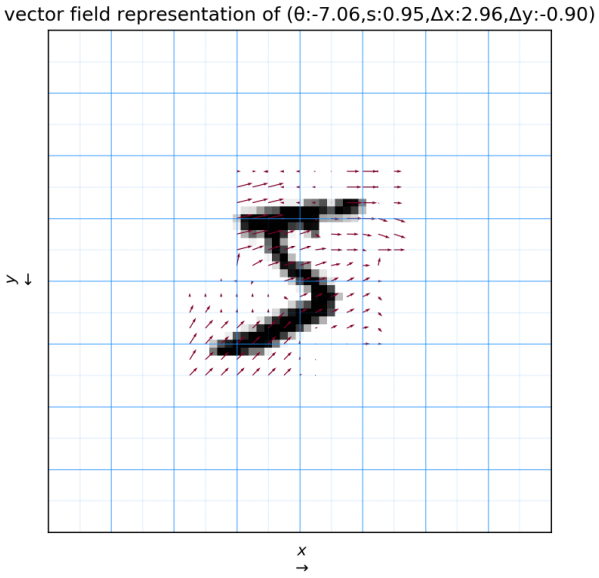

Figure 2.12: Visualization of local shifts resulting from a global affine transformation featuring translation, rotation and scale.

transform model for video prediction. Before we move on to the prediction step, we believe that it is important to verify the validity of the procedure so far. This is accomplished by ensuring that $\mathcal{PD}_{u,v}$ is interpretable in the sense that it provides a meaningful description of the true image transform. For example, looking at Figure 2.12, it is not obvious that the displayed vector field results from an affine transform that rotates the digit by $-7°$, shrinks it by 5% and shifts it along both axes, so it is natural to ask whether this is information is preserved by our preprocessing.

2.3.1. Experiment Design

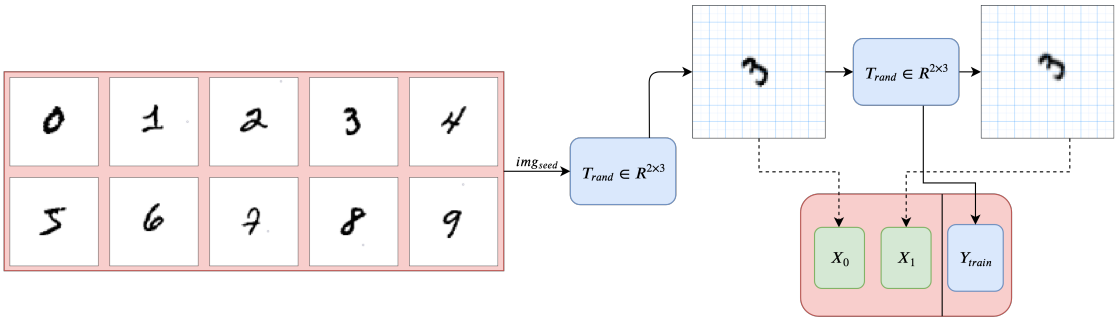

Figure 2.13: Setup of the experiment.

Therefore, we design the dataset depicted in Figure 2.13. Each sample is generated by randomly selecting 1 out of 10 different MNIST digits (LeCun et al. [27]). The digit is then oriented randomly to produce the seed frame x_0. Next, an affine transformation T with random rotation, scale and shifts is applied to the seed frame, yielding x_1. The tuple (x_0, x_1, T) is then appended to the dataset.

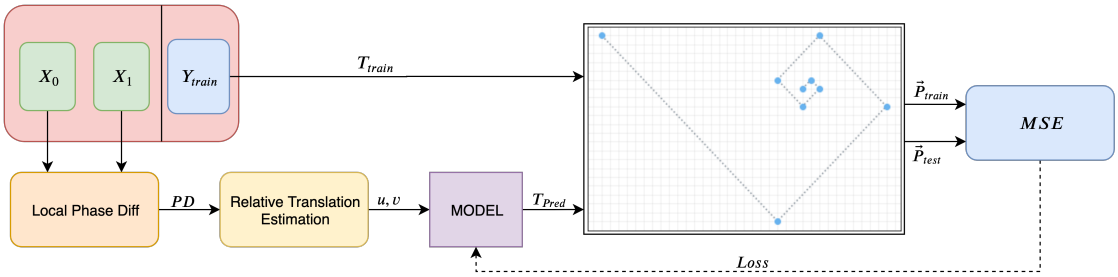

Figure 2.14: Overview of the training process.

To train a model to estimate T, the LFT is applied to x_0 and x_1. From the results, $\mathcal{PD}_{u,v}$ is calculated and then mapped to local translation by $\text{softargmax}(\text{pd}_{u,v}) = (\Delta y, \Delta x)$. The network is then presented exclusively with $(\Delta y, \Delta x)$ and predicts the entries of the 2×6 matrix T. Since the matrix entries for rotation, scale and shifts have very different effects, simply calculating the mean square error between the ground truth transformation and the predicted one is problematic, even though this is applied in a different context by Amersfoort et al. [7]. Instead, we create a test point set consisting of the first nine points at the edges of the rectangles commonly used to approximate the golden spiral, as they are placed at diverse distances from the center of rotation and scale at the coordinate system origin. These are mapped by both the network output transformation and the ground truth transformation. The MSE between the two resulting point sets is then used to adjust the network weights via backpropagation. This training process is summarized in Figure 2.14.

Some results are shown in Figure 2.15, with one example above, near and below the average validation loss of ≈ 1.1. Here, the estimated affine transformation is applied to the seed image to visualize how close the predicted transformation is to the ground truth. It is noteworthy that the MSE between the entries of the predicted affine transformation and the ground truth affine transformation decreases as intended along with the training loss, even though the training loss does not directly target these values.

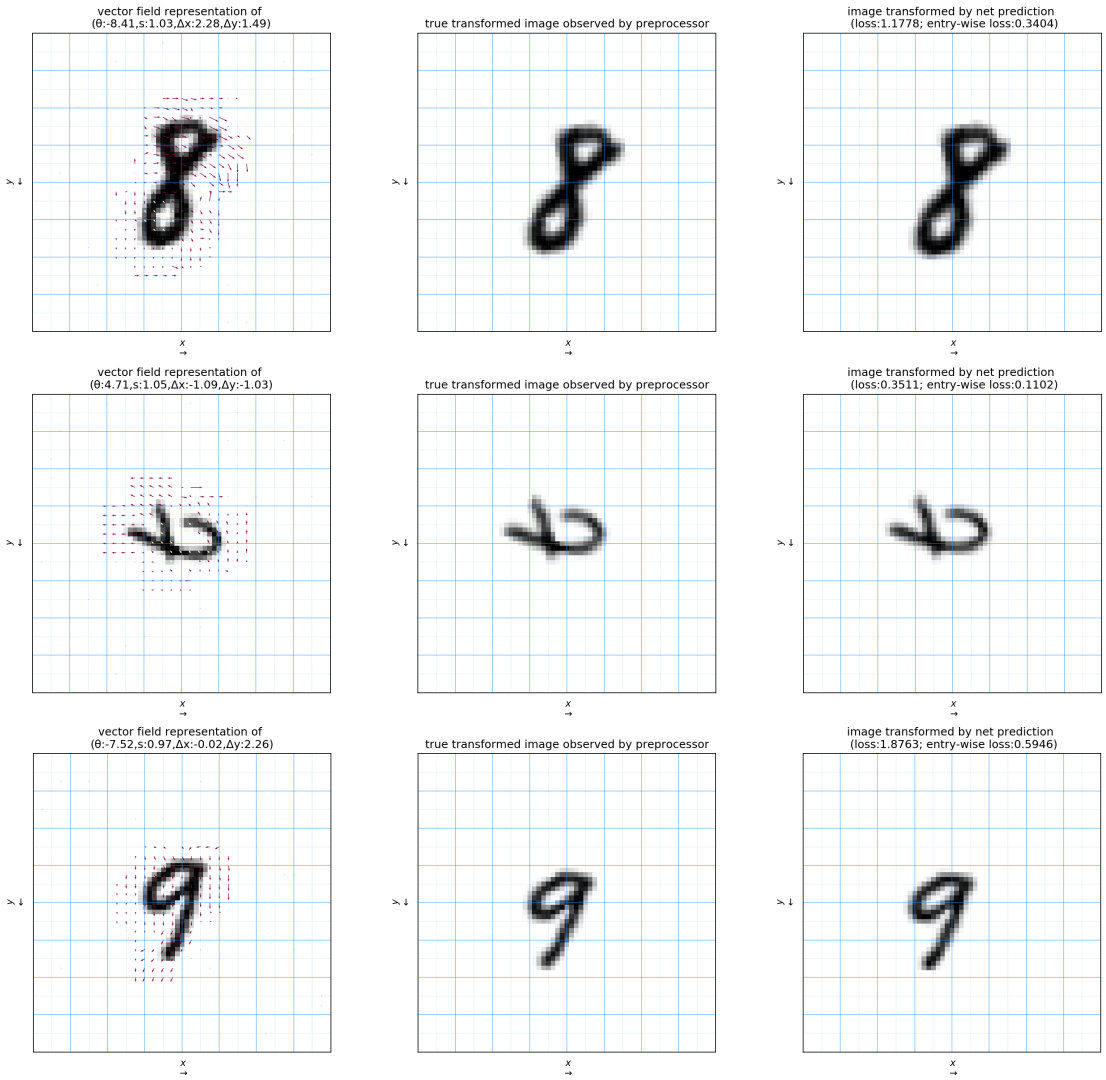

Figure 2.15: *From top to bottom*: an affine transformation estimate of average, high and low quality from the validation set (by comparison to the average validation loss).

3. Synthesis

3.1. Prediction by Local Phase Addition

In the previous chapter, we covered the extraction of the local phase differences $\mathcal{PD}_{t,u,v}$ between two frames x_{t-1} and x_t under the assumption that x_t is the result of local circular shifts applied to x_{t-1}.

If the transformation from x_t to x_{t+1} remains constant, applying equation (2.3) yields that the Fourier partners of the local views on x_{t+1}, can be calculated by the element-wise product

$$\mathcal{X}_{t+1,u,v} = \mathcal{X}_{t,u,v} \cdot \mathcal{PD}_{t,u,v}. \tag{3.1}$$

This corresponds to adding the phase differences in $\mathcal{PD}_{t,u,v}$ to the phases in each frequency bin in $\mathcal{X}_{t,u,v}$. Therefore, we refer to this operation as *local phase addition*. In practice, zero-padding of $x_{t,u,v}$ with a *padding size* pS prevents wrap-around effects that can occur at this stage, as proposed by Allen [28]. It is safely pruned further down the line, but until then, the cells feature a side length of $N' \coloneqq N + 2pS$.

Utilizing the inverse FFT (iFFT), cells covering x_{t+1} are recovered as

$$\tilde{x}_{t+1,u,v}[n,m] = \frac{1}{N^2} \sum_{\omega_1}^{N-1} \sum_{\omega_2}^{N-1} \mathcal{X}_{t+1,u,v}[\omega_1, \omega_2] e^{j\frac{2\pi}{N}(\omega_1(n-uH)+\omega_2(m-vH))} \tag{3.2}$$

$$uH \leq n \leq uH + N - 1 \tag{3.3}$$

$$vH \leq m \leq vH + N - 1 \tag{3.4}$$

.

Note that we do not refer to these local predictions as $x_{t+1,u,v}$. Our intention is to highlight that the description of the image transformation relies on idealized conditions. Presupposing *geotemporal permanence* of the local circular shifts can be described by

$$(\Delta y_{GT}, \Delta x_{GT})_{t+1,u,v} \approx (\Delta y, \Delta x)_{t,u,v} \tag{3.5}$$

where GT refers to the true local image shifts. In reality, objects in the scene experience innate motility, and the corresponding local transformations can *move along with them*. This is more in line with

$$(\Delta y_{GT}, \Delta x_{GT})_{t+1,u+\Delta_H y,v+\Delta_H x} \approx (\Delta y, \Delta x)_{t,u,v} \tag{3.6}$$

notwithstanding higher order derivatives of the image transformation and the fact that $\mathcal{PD}_{t,u,v}$ is generally noisy. The discrepancy between equations (3.5) and (3.6) emerges as the most severe source of prediction errors, and is treated in its own chapter.

3.2. From Local to Global Prediction

Calculating $\tilde{x}_{t+1,u,v}$ yields local predictions, but they are modified by the extraction window w, rendering the recovery of the global prediction x_{t+1} non-trivial. Perfect reconstruction of x_{t+1} given $\tilde{x}_{t+1,u,v}$ and w is therefore the primary objective of this chapter. Considering $\tilde{x}_{t+1,u,v}$ is the result of the inverse DFT of $\mathcal{X}_{t+1,u,v}$ the process of perfect reconstruction is equivalent to inverting the LFT.

3.2.1. Overlap-Add Equations

In analogy to the 1D case (Sharpe, Li [14, 15]), regard to this end

$$\begin{aligned}&\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}\tilde{x}_{t+1,u,v}[n,m]w^a[n-uH,m-vH]\\&=x_{t+1}[n,m]\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}w^{a+1}[n-uH,m-vH].\end{aligned}$$

The derivation is shown in appendix A. Rearranging the first and the last terms yields

$$x_{t+1}[n,m]=\frac{\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}\tilde{x}_{t+1,u,v}[n,m]w^a[n-uH,m-vH]}{\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}w^{a+1}[n-uH,m-vH]} \tag{3.7}$$

the *overlap-add equation* for the LFT at the core of the inverse Local Fourier Transfrom (iLFT). In our implementation, we set $a = 1$ to avoid having to interpret potential terms including 0^0 in the numerator of equation (3.7). In the context

of the STFT, $a = 1$ also yields an optimal reconstruction in a least squares sense (Griffin et al. [29]), but we have not confirmed that this holds for the LFT. In practice, the overlap-add equation is implemented by transforming the supports of the addends to a common coordinate system and adding them to a two dimensional buffer with adequate overlap determined by the hop size. Taking into account the extraction described in 2.2.1, the padding of size $\lfloor \frac{N}{2} \rfloor$ must be cropped. Special care is necessary to preserve gradients through this step. Since the local cells can be added to the buffer independently, the overlap add computation was optimized via vectorization.

3.2.2. Window Function Constraints

In analogy to the 1D case and with regards to the denominator in (3.7), we constrain for w and all n, m one of

$$\sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} w^2[n - uH, m - vH] = 1(\text{Strong COLA}) \tag{3.8}$$

$$\implies \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} w^2[n - uH, m - vH] = C(\text{COLA}) \tag{3.9}$$

$$\implies \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} w^2[n - uH, m - vH] \neq 0(\text{NOLA}) \tag{3.10}$$

Here, only the *Nonzero Overlap Add* (NOLA) constraint is a necessary condition.

However, there is a common misunderstanding that *Constant Overlap Add* (COLA) is a necessary condition. As evident by the derivation above, this is in fact untrue. This misconception is indeed puzzling, as Reference (Griffin et al. [29]) was published in 1984 and presents the same inversion rule as Reference (Röbel [30]) and above for the case $a = 1$. Nevertheless, *SciPy* offers a check_COLA function (Scipy [31]) that was used to assert correct inversion of the *STFT* until Sharpe [14] resulted in replacing this by check_NOLA. Moreover, there are examples of this in the literature, for example in Reference (Smith [32]) even strong COLA is demanded to hold for the window functions used. This of course greatly simplifies the inversion rule, at the cost of restricting the range of possible windows to use.

For the LFT, we cannot afford to demand COLA, as it restricts the available window functions to exclusively the rectangle window. This is because 2D COLA is much more restrictive than COLA in 1D.

This is depicted in Figure 3.1, where individual windows are drawn in dashed

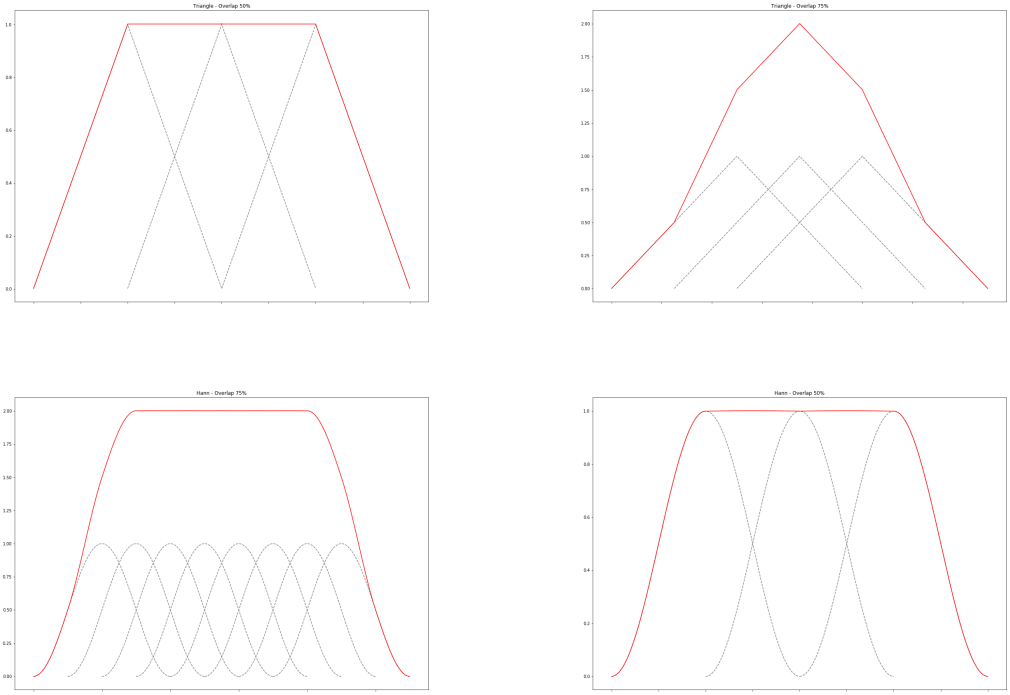

Figure 3.1: Various examples of 1D COLA. The triangle window at 50% overlap (*top-left*, strong COLA holds), the triangle window at 75% overlap (*top-right*, COLA does not hold), the Hann window at 75% overlap (*bottom-left*, COLA holds), the Hann window at 50% overlap (*bottom-right*, strong COLA holds).

gray, and their sum in red. In the top left, strong COLA holds, as the values of the triangle windows sum up to exactly unity when the windows are placed apart to have an overlap of 50%. It is important to note that this is a property of the window function together with the selected overlap. Regard the triangle window at 75% overlap in the top right, where COLA does not hold. Other window functions satisfy COLA at several different overlaps (Heinzel et al. [33]). In the case of the Hann window, COLA is satisfied at an overlap of 75%, as seen in the bottom left and 50% as shown in the bottom right. A full discussion of the COLA constraint for many different commonly used 1D windows can be found in Reference (Heinzel et al. [33]), where it is referred to as *amplitude flatness*.

In the 2D case, the peaks of window functions covering a certain cell are not located at an equal distance to the peaks of windows covering adjacent, overlapping cells. This is because the neighboring cells in the diagonal direction are located further away by a factor of $\sqrt{2}$ than neighboring cells in axis-parallel directions. On the other hand, the value of the window function depends only on the distance

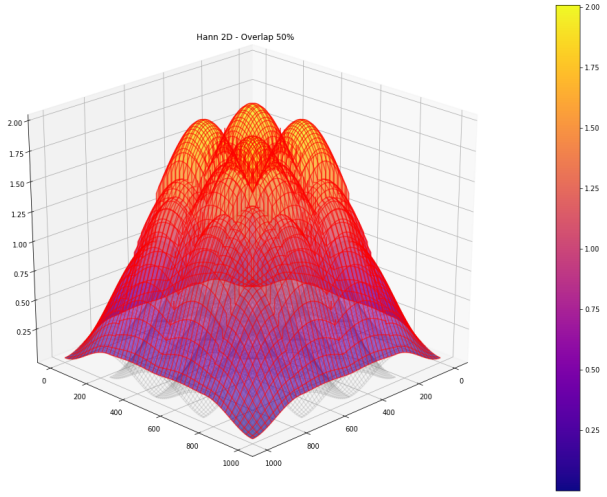

Figure 3.2: 2D Hann windows at 50% overlap. Unlike in the 1D case, COLA does not hold.

from the center of the corresponding cell, as the 2D windows are radialized 1D windows. This results in a repeating, bumpy pattern in the amplitude of the sum over all windows for all but the rectangle window function. As an example, the 2D Hann window is plotted in Figure 3.2 arranged on a 3×3 grid of with an overlap of 50%. It is evident that the sum over these windows does not satisfy COLA.

Unfortunately, my initial assessment was that the perfect reconstruction of an LFT is impossible due to this strict constraint. We developed an alternative reconstruction method by arranging the predicted local cells in a grid and convolving this grid with a dilated kernel. The kernel has nonzero entries only where the entries of neighboring cells are located relative to the pixel that the kernel is anchored on. As long as the kernel entries are normalized, this convolution computes a weighted average for an individual pixel in x_{t+1} over all contributions for that pixel in $x_{t+1,u,v}$. This does not account for the windowing function and thus, a high error incurs, thus we discard this approach.

For the reconstruction using (3.7), after extracting cells along the border of the image in the LFT as per 2.2.1, we only need to respect NOLA, and this only within the confines of the image, not the padding added for valid cell extraction. This way, only pathological windows with minuscule support subordinate to the hop size prevent the inverse LFT using the overlap add equation. As long as NOLA holds, the error for consecutive LFT and iLFT on an image is in the same order of

magnitude as for an FFT and iFFT pair, independently of hop size and window used.

3.3. Overview of Previous Steps

To provide an overview of the discussion so far, we concentrate the most important steps of the prediction process.

Algorithm 1: Local Fourier Transform - LFT

Data: batch of images x, shaped $B \times U \times V$,
window function w, shaped $N \times N$
Parameters: hop size H, padding size pS
Result: batch of LFT results $\mathcal{X}_{u,v}$, shaped $B \times L \times N' \times N' \times 2$
$x_{u,v} \leftarrow$`extract_local_windows(`x, N, H`)`
$x_{u,v} \leftarrow x_{u,v} \cdot w$
$x_{u,v} \leftarrow$`zero_pad(`$x_{u,v}$, pS`)`
$\mathcal{X}_{u,v} \leftarrow$`FFT(`$x_{u,v}$`)`
return $\mathcal{X}_{u,v}$

Algorithm 2: get_phase_differences

Data: batch of LFT results $\mathcal{X}_{t-1,u,v}$, shaped $B \times L \times N' \times N' \times 2$,
batch of LFT results $\mathcal{X}_{t,u,v}$, shaped $B \times L \times N' \times N' \times 2$
Parameters: stabilizer for weak denominator ε
Result: batch of phase differences $\mathcal{PD}_{t,u,v}$, shaped $B \times L \times N' \times N' \times 2$
$\mathcal{PD}_{t,u,v} \leftarrow \frac{\mathcal{X}_{t,u,v}\overline{\mathcal{X}_{t-1,u,v}}}{|\mathcal{X}_{t,u,v}\overline{\mathcal{X}_{t-1,u,v}}|+\varepsilon}$
return $\mathcal{PD}_{t,u,v}$

Algorithm 1 describes the calculation of the LFT and algorithm 2 the extraction of local phase differences given two LFTs, thereby summarizing chapter 2.

Algorithm 3: phase_add

Data: batch of LFT results $\mathcal{X}_{t,u,v}$, shaped $B \times L \times N' \times N' \times 2$,
batch of phase differences $\mathcal{PD}_{t,u,v}$, shaped $B \times L \times N' \times N' \times 2$
Result: batch of LFT results $\mathcal{X}_{t+1,u,v}$, shaped $B \times L \times N' \times N' \times 2$
$\mathcal{X}_{t+1,u,v} \leftarrow \mathcal{X}_{t,u,v} \cdot \mathcal{PD}_{t,u,v}$
return $\mathcal{X}_{t+1,u,v}$

Algorithm 3 and 4 recapitulate sections 3.1 and 3.2.1, whilst algorithm 5 combines all subtasks, defining the complete prediction process.

Algorithm 4: inverse Local Fourier Transform - iLFT

Data: batch of LFT results $\mathcal{X}_{u,v}$, shaped $B \times L \times N' \times N' \times 2$,
window function w, shaped $N \times N$
Parameters: hop size H, padding size pS
Result: batch of images x, shaped $B \times U \times V$
$\tilde{x}_{u,v} \leftarrow$`iFFT(`$\mathcal{X}_{u,v}$`)`
$\tilde{x}_{u,v} \leftarrow$`crop(`$\tilde{x}_{u,v}$`, `$pS$`)`
$\tilde{x}_{u,v} \leftarrow \tilde{x}_{u,v} \cdot w$
$num \leftarrow$`overlap_add(`$\tilde{x}_{u,v}$`, `H`)`
$denom \leftarrow$`overlap_add(`w^2`, `H`)`
$x \leftarrow \frac{num}{denom}$
$x \leftarrow$`crop(`x`, `$\lfloor \frac{N}{2} \rfloor$`)`
return x

Algorithm 5: predict_next_frame

Data: batch of images x_{t-1}, shaped $B \times U \times V$
batch of images x_t, shaped $B \times U \times V$,
window function w, shaped $N \times N$
Parameters: hop size H, padding size pS
Result: batch of images x_{t+1}, shaped $B \times U \times V$
$\mathcal{X}_{t-1,u,v} \leftarrow$`LFT(`$x_{t-1}$`, `$w$`)`
$\mathcal{X}_{t,u,v} \leftarrow$`LFT(`$x_t$`, `$w$`)`
$\mathcal{PD}_{t,u,v} \leftarrow$`get_phase_differences(`$\mathcal{X}_{t,u,v}$`, `$\mathcal{X}_{t-1,u,v}$`)`
$\mathcal{X}_{t+1,u,v} \leftarrow$`phase_add(`$\mathcal{X}_{t,u,v}$`, `$\mathcal{PD}_{t,u,v}$`)`
$x_{t+1} \leftarrow$`iLFT(`$\mathcal{X}_{t+1,u,v}$`, `w`)`
return x_{t+1}

3.4. Greedy multi-step Prediction

Up to now, we described a process to predict video frames one time step into the future. Of course, we are interested in predictions over a further time horizon.

This is not straight-forward, because the movement estimation is local and thus only valid while it is located within the receptive field of the corresponding cell. This is shown in Figure 3.3, where the local shifts at time t are superimposed on the position of the digit at time $t + 4$, revealing violations of the geotemporal permanence assumption introduced in section 3.1.

The consequence of applying a prediction in this manner is showcased in Figure 3.4. The prediction of the transformation is incorrect, and the error accumulates to an extent that the digit dissolves.

This issue is not as pronounced when the windows are large, and in the extreme

Figure 3.3: The frame x_t along with the transformation towards x_{t+1} superimposed in red (*left*). The frame at a later time x_{t+4}, showing that the local predictions are no longer valid, a clear violation of the assumption of geotemporal permanence (*right*).

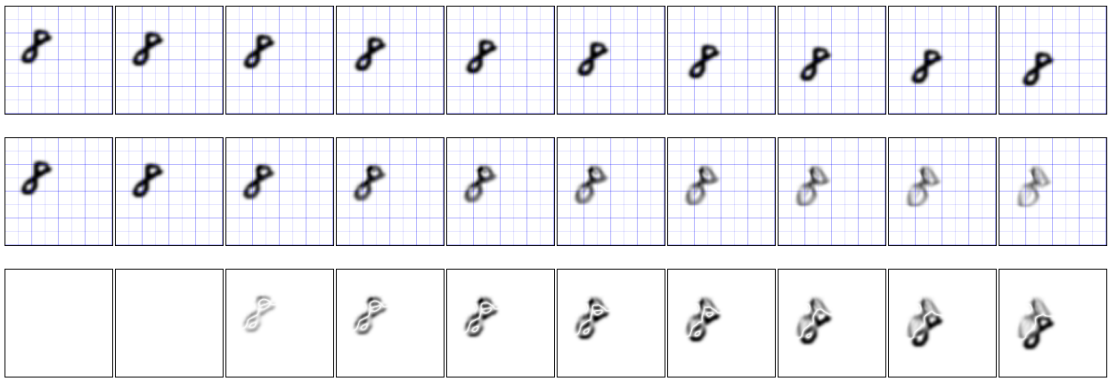

Figure 3.4: A digit dissolving over several prediction steps due to unaddressed violations of geotemporal permanence at a small window size (7×7). *From top to bottom*: ground truth sequence of length 10, 2 seedframes and 8 predictions, difference images.

case of windows covering the whole image, shown in Figure 3.5, does not cause prediction to fail. On the other hand, we cannot adequately resolve more complex global transformations using large windows, so increasing the cell sizes to alleviate this problem is contrary to our goals.

One alternative is to predict next frames in a greedy manner. We iterate predicting the next frame, calculating a new transformation based on the most recent prediction, and using that to update the prediction to the next time step. This process is diagrammed in Figure 3.6.

In practice, transformations predicted in this manner will decelerate while the moving object experiences a characteristic drain of image intensity starting on the side opposite to the direction of movement, as evident in Figure 3.7. The process is clearly not stable, and susceptible to accumulating errors. However, the structure of moving objects remains intact, and the defects in the resulting prediction appear systematic, implying that refinement is possible.

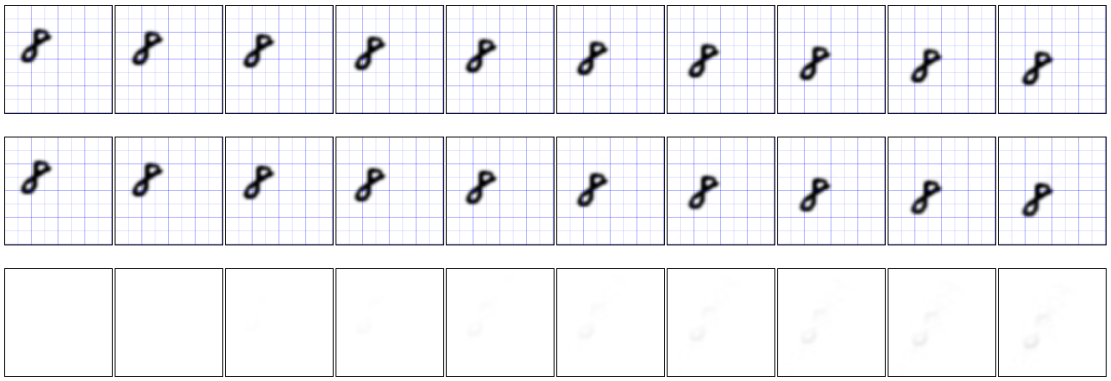

Figure 3.5: Prediction sequence for very large windows covering the whole image. *From top to bottom*: ground truth sequence of length 10, 2 seedframes and 8 predictions, difference images.

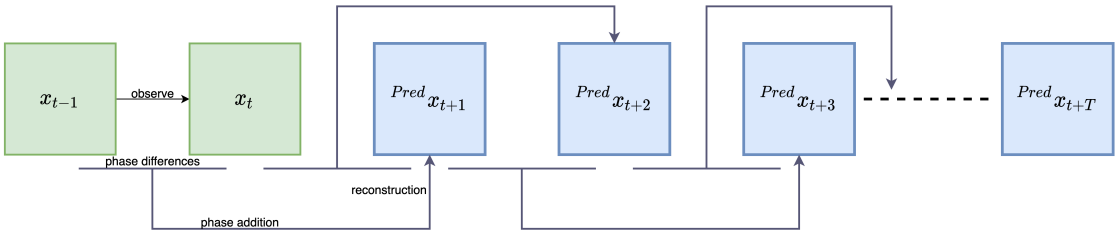

Figure 3.6: The greedy multi-step prediction scheme.

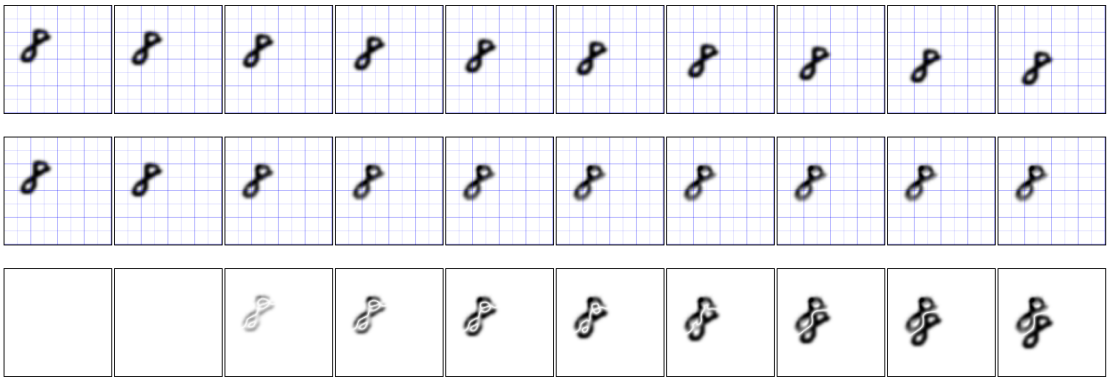

Figure 3.7: The typical effect of applying greedy prediction. *From top to bottom*: ground truth sequence of length 10, 2 seedframes and 8 greedy predictions, difference images.

Notably, the inevitable buildup of errors accompanying this prediction scheme introduces the necessity to filter outliers in the local prediction, as exemplified in Figure 3.8, where the large local shifts pointing opposite to most neighboring values are plausible only within their own cells. From the point of view of the misjudging cells, the lower bar of the digit 1 enters the cell at the bottom, just as the upper cell

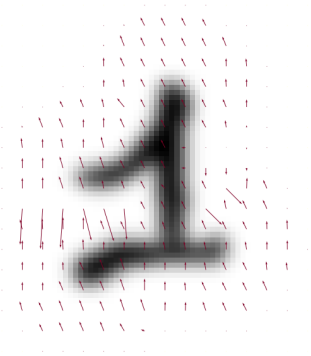

Figure 3.8: An example of a common local misjudgment type.

contents disappear from its view. Consequently, the local translation is estimated to be a large downward shift, while small upwards shifts are more plausible when taking into account the other nearby shift estimates.

In conclusion to this chapter, we stress that refinement is unavoidable due to breaches of geotemperoal permanence as well as hallucinated local shifts. We introduce the prediction process as shown in 3.6 and upon this base the methods proposed in the following chapter.

4. Refinement

In the previous chapter, we established the necessity of a refinement that addresses breaches of geotemporal permanence as well as local misadventures. Before this can be developed further, we must secure a dataset of toy examples that feature individual objects rotating, shifting and changing their scale.

4.1. Dataset Generation

Figure 4.1: An image deteriorating as a result of accumulating interpolation errors at low resolution affine warps.

The primary concern in this regard is the culmination of interpolation errors common to low resolution image transformation. To appraise the resulting errors, we set up a basic trial. An image from the MNIST dataset (LeCun et al. [27]) is selected, and mapped via a series of rotations that, in theory, amount to identity. Specifically, we apply 45 rotations of 8°. At the original resolution of 34×34, the image quality gradually deteriorates until the digit is barely legible, showcased in Figure 4.1.

This chain of transformations amounting to identity is represented by the red arrow in Figure 4.2, and the rows marked with the corresponding color in table 4.1 summarizes the discrepancy in terms of the pixel-wise L1 loss. We also upsample the digit K times, each operation doubling the image size along each axis. With increasing resolution, the interpolation errors vanish, shown in the rows marked in blue. Nevertheless, mapping the digit through several upsample steps, performing the rotations, and then finally downsampling again also causes an error, stated in yellow. This is is explained by the fact that simply up- and then downsampling also causes errors that converge as K increases, displayed in black. In consequence, the only viable solution to generate ground truth data that is itself not inherently

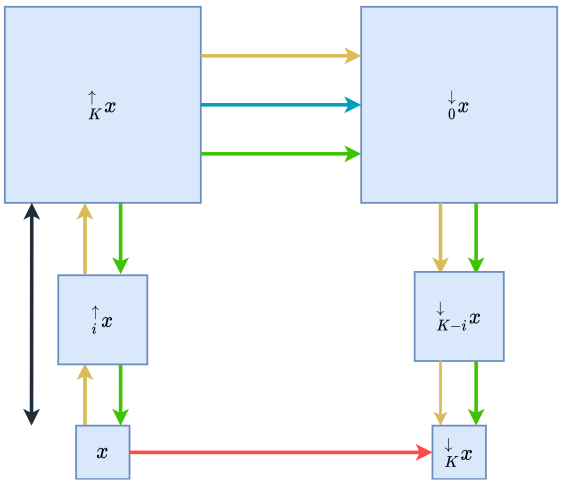

Figure 4.2: An overview of the dataset generation errors described in this chapter.

$\uparrow K$	$K=1$	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$
Mean Error	3.44e−2	3.93e−2	4.04e−2	4.06e−2	4.07e−2	4.07e−2	4.07e−2	4.07e−2
Maximum Error	4.84e−1	5.26e−1	5.35e−1	5.37e−1	5.38e−1	5.38e−1	5.38e−1	5.38e−1
Mean Error	9.08e−2	9.08e−2	9.08e−2	9.08e−2	9.08e−2	9.08e−2	9.08e−2	9.08e−2
Maximum Error	8.16e−1	8.16e−1	8.16e−1	8.16e−1	8.16e−1	8.16e−1	8.16e−1	8.16e−1
Mean Error	3.81e−2	1.30e−2	3.60e−3	9.00e−4	2.00e−4	5.93e−5	1.48e−5	3.70e−6
Maximum Error	4.53e−1	1.73e−1	5.22e−2	1.36e−2	3.40e−3	9.00e−4	2.00e−4	1.00e−4
Mean Error	6.50e−2	4.89e−2	4.30e−2	4.13e−2	4.09e−2	4.08e−2	4.07e−2	4.07e−2
Maximum Error	7.03e−1	5.99e−1	5.55e−1	5.42e−1	5.39e−1	5.38e−1	5.38e−1	5.38e−1
Mean Error	3.18e−2	9.90e−3	2.70e−3	7.00e−4	2.00e−4	4.30e−5	1.07e−5	2.73e−6
Maximum Error	3.60e−1	1.20e−1	3.27e−2	8.30e−3	2.10e−3	5.00e−4	1.00e−4	3.56e−5

Table 4.1: Overview of Errors from different sources.

tainted by errors is to upsample the digits, discard the low resolution versions, calculate the sequence of transformations in a high resolution, and then downsample the whole sequence. The resulting errors are negligible for $K \geq 7$, as listed in the green row. An example sequence containing several snapshots of a digit rotating is pictured in Figure 4.3.

Figure 4.3: An image sequence transformed at a high resolution multiple times, and subsequently downsampled.

Overall, the compilation of a dataset requires the following steps:

1. Select two random MNIST digits.
2. Upsample the images $K = 7$ times to a resolution of $4352px \times 4352px$.
3. Per digit, choose an initial orientation at random.
4. Uniformly sample for each a scale change from $[-1\%, +1\%]$ and a rotation from $[-7°, 7°]$.
5. Apply the transformations to the respective images 9 times, yielding two sequences of length 10.
6. Downsample the sequences down to $136px \times 136px$.
7. Place the sequences into a common $260px \times 260px$ canvas, each subsequent frame translated by a random uniform integer amount from $[-8, 8]$.
8. Downsample the canvasses to $65px \times 65px$, simulating subpixel shifts.

4.2. Evaluation of Prediction Quality

To encourage improved prediction quality, the refinement process must be provided an adequate objective function. The goal is to formulate an error metric which, when optimized, aligns with increasing perceived visual quality of the predicted frame. In this regard, simply using a mean square error is insufficient, as blurry predictions are not penalized in a satisfactory manner (Amersfoort et al., Svoboda et al. [7, 34]).

Instead, we rely on the *Structural SIMilarity* (SSIM) Index defined by Zhou Wang et al. [35] as

$$\mathrm{SSIM}(x, x_{pred}) \coloneqq \frac{(2\mu_x\mu_{x_{pred}} + C_1)(2\sigma_{x,x_{pred}} + C_2)}{(\mu_x^2 + \mu_{x_{pred}}^2 + C_1)(\sigma_x^2 + \sigma_{x_{pred}}^2 + C_2)} \tag{4.1}$$

where μ_x and σ_x are estimates of the mean intensity and standard deviation within an image region, respectively. Furthermore, $\sigma_{x,x_{pred}}$ is the correlation coefficient estimated by

$$\sigma_{x,x_{pred}} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(x_{pred,i} - \mu_{x_{pred}}) \tag{4.2}$$

and C_1, C_2 stabilize the division with weak denominators. This is viewed as a loss function called the *Structural Dissimilarity* (DSSIM) (Riba et al. [36]) by

$$\mathrm{DSSIM}(x, x_{pred}) = \frac{1 - \mathrm{SSIM}(x, x_{pred})}{2} \tag{4.3}$$

.

Like SSIM, the DSSIM loss is calculated between small kernels sliding over both inputs, effectively providing a differentiable local comparison of luminance, contrast and structure. Specifically, we use a GPU-accelerated implementation provided in *Kornia* (Riba et al. [36]) with a window size of 5×5. To tackle the drain of image intensity described in Chapter 3 directly, we define the prediction loss as an interpolation between DSSIM and L1 losses, both reduced to their mean values

$$Loss_{pred}(x, x_{pred}) \coloneqq \alpha\, \mathrm{DSSIM}_{avg}(x, x_{pred}) + (1 - \alpha)|x - x_{pred}|_{avg} \tag{4.4}$$

.

In practice, we set $\alpha = 0.5$ and observe that $Loss_{pred} < 5\mathrm{e}{-3}$ corresponds to a satisfactory prediction, while $Loss_{pred} < 8\mathrm{e}{-4}$ describes predictions that are visually indistinguishable from the ground truth.

4.3. Pre- versus Post-prediction Refinement

Since the whole pipeline described in this thesis is differentiable, we can, in principle, train a model to improve the output at each stage of video prediction. However, in terms of finding an appropriate point of application, there are two intuitive candidates to refine the prediction. First, we could adjust the prediction after it has been created. i.E using a learned residual image. The alternative would be to refine the phase differences before they are added to the LFT of the current frame.

Our initial experiments evaluated the former approach. We adapted the L8-Net described by Svoboda et al. [34] as a tool to correct image degradation resulting from JPEG compression. It refers to a fully convolutional model consisting of 8 layers with kernel sizes varying between 11×11 and 1×1. To facilitate the propagation of information through these layers, a *skip architecture* (Shelhamer et al. [37]) passes a copy of the outputs of higher layers to lower layers, bypassing intermediate layers. At the lower layers, the previous output is concatenated to the input, providing geometric information at that stage. Like the original L8 net, we pass the output of the first layer to the fourth and sixth layers. We change

this model by not only presenting it with the predicted frame, but also with an estimate of the local linear shifts upsampled to match the image resolution. This information is generated by the soft argmax described in Chapter 2 and passed in two additional input channels. In turn, the model outputs a residual image that is added to the prediction to correct errors. Unfortunately, the model would not train when we zeroed out the image intensity channel from the orignal prediction, and improved just as well when we zeroed the channels for the local shifts along y and x directions. While we are able to correct predictions in this manner, our conclusion is that the model has learned the semantics of the MNIST digit data and is using that information to correct the frames, instead of taking into account the extracted local transformations. This is possible because the receptive field of the L8 net is $25px \times 25px$ (Svoboda et al. [34]), so it would fit a whole digit. It is also undesirable, as it is unlikely that such information is useful for natural images.

Therefore, we warn against correcting the prediction using a residual image based on the prediction, and propose a method for refining $\mathcal{PD}_{u,v}$ before it is used to update the current frame.

4.4. Proposed Architecture

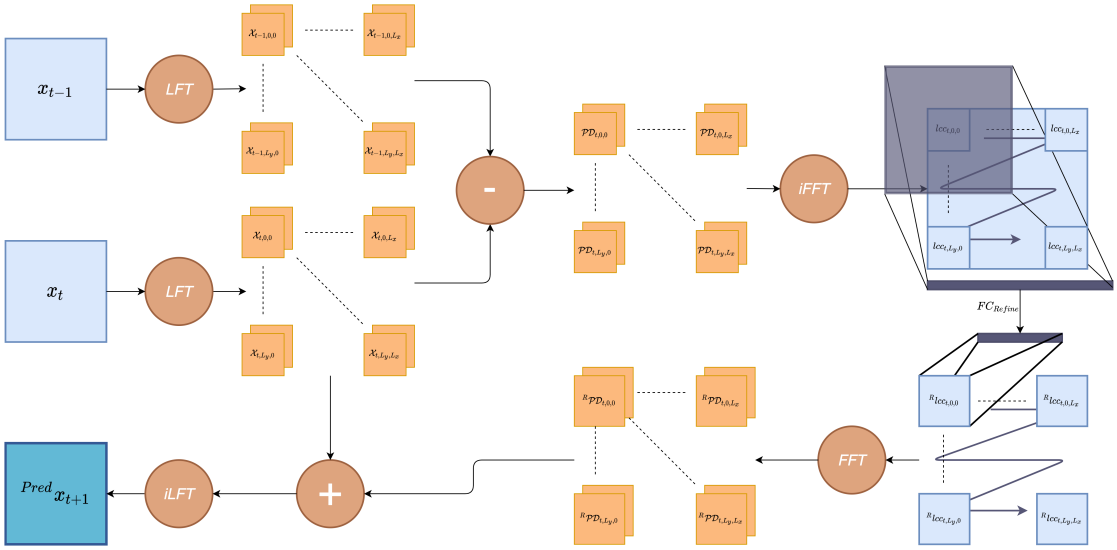

Figure 4.4: The proposed pre-prediction refine model.

Building up on the results of the previous section, we attempt to decouple any efforts at improving the prediction from the semantics of the sequence dataset. This is achieved after the extraction of $\mathcal{PD}_{u,v}$ which encodes only local translation,

but is nescient regarding image contents. As depicted in Figure 4.4, we transform $\mathcal{PD}_{t,u,v}$ to local cross-correlation matrices in spatial domain, represented by $lcc_{t,u,v}$, via iFFT. We view the collection of these terms as lying on a grid with contiguous axes, appropriately zero-padded, then slide a square window over the grid. This operation has a stride of N', thus the window is always anchored on the center of a cross-correlation matrix. To cover exactly all cells at a Chebyshev distance, in terms of N', of less than K, the window side length is defined as $(2K+1)N'$. For example, for $K = 2$, the window contains a 5×5 grid of terms $lcc_{t,u,v}$. This is then flattened and transformed by one fully connected layer to a vector of length N'^2, which is then reshaped to $N' \times N'$, yielding ${}^{R}lcc_{t,u,v}$ and replacing the cross-correlation that the window was centered on. By sliding over all terms in this manner, sharing the parameters of the fully connected layer between all extraction locations, the transport of information between encodings of local shift is encouraged within a neighborhood defined by K. Of course, the number of model parameters is quartic as a function of N', but for our windows of size 7 to 15, this is tractable, the model parameters then occupy about 5 MB. Finally, ${}^{R}lcc_{t,u,v}$ is mapped to ${}^{R}\mathcal{PD}_{t,u,v}$ by FFT, which in turn, is used to predict the next frame ${}^{Pred}x_{t+1}$.

In the context of training and evaluation, the pipeline segments presented in 4.4 leading up to the refine model input are handled by a preprocessor, and the steps following the network output up to the prediction of the next frame by a postprocessor. These share access to $\mathcal{X}_{t,u,v}$.

The preprocessor observes the first seed frame ${}^{GT}x_{t-1}$ and caches the LFT $\mathcal{X}_{t-1,u,v}$ of x_{t-1}. Then it observes the second seed frame ${}^{GT}x_t$. It calculates $\mathcal{X}_{t,u,v}$, and since $\mathcal{X}_{t-1,u,v}$ is cached, obtains the phase differences $\mathcal{PD}_{t,u,v}$. In the process, the cached terms are overwritten by $\mathcal{X}_{t,u,v}$. Then $\mathcal{PD}_{t,u,v}$ is transformed to $lcc_{t,u,v}$ and passed to the refine model, which outputs the adjusted terms ${}^{R}lcc_{t,u,v}$. The postprocessor calculates the refined phase differences ${}^{R}\mathcal{PD}_{t,u,v}$ and applies them to $\mathcal{X}_{t,u,v}$. Finally, it uses the iLFT to obtain ${}^{Pred}x_{t+1}$. This is then passed to the preprocessor, where the process begins anew, until, for a fixed time horizon T, a set $\{{}^{Pred}x_{t+1}, \ldots, {}^{Pred}x_{t+T}\}$ of predictions is gathered. Consequently, the loss is calculated with respect to $\{{}^{GT}x_{t+1}, \ldots, {}^{GT}x_{t+T}\}$. Of course, the loss corresponding to the predicted frame at time $t+i$ depends on the quality of the predictions for the frames $\{{}^{Pred}x_{t+i-1}, \ldots, {}^{Pred}x_{t+1}\}$. Therefore, at the end of each epoch, the chain of predictions is unrolled and the loss backpropagated through time (Mozer [38]), taking into account previous prediction errors. Since a prediction loss is not meaningful if the previous predictions were poor, the training process runs through a warm-up phase, during which the prediction time horizon $k \in \{1, \ldots, T\}$ is increased based on the training epoch number. This way, the refine model is also

exposed to the synthetic frame predictions that it produces, while getting accustomed to the accumulating prediction error in a gradual manner.

Figure 4.5: Vector fields visualizing $lcc_{u,v}$ before (*left*) and ${}^{R}lcc_{u,v}$ after (*right*) refinement.

After training, the refine model has learned to adjust the estimated local shifts by increasing the confidence where neighboring cells have similar values, and smoothing outliers. This can be observed in Figure 4.5, where the an estimate of the local transformation is shown before and after refinement. The vectors drawn using the soft argmax with $\tau = 0.02$, are clearly longer for the refined local cross-correlations, reflecting a higher confidence in the extracted translation. Also, the border regions of the vector field have been smoothed, resulting in what can be compared to a morphological dilation. However, it is also evident that the velocities at several locations are not plausible, especially for the digit 2. This shows that in this example, which is from the validation set, the refinement has not yet learned to fix all kinds of failure cases. However, since the window size used here was 9×9, we do not expect a handful of misjudgments viewing only a small section of the digit to destabilize the prediction. Also, the results imply that the output of the model can still be interpreted as a local cross-correlation, which is not clear a priori as the loss function evaluates only the prediction quality.

4.5. Differentiable Window Function Selection

Currently, the local cells are small, and consequently, the window functions are represented by only a few samples. For high resolution application, it is plausible that the local cells can be larger in terms of their pixel size. The window functions would then be of sufficient size to render spectral analysis meaningful. Especially, it should be possible to fine-tune the windowing to the pixel velocities present in the scene. This is described for Gabor Kernels useful to optical flow estimation in

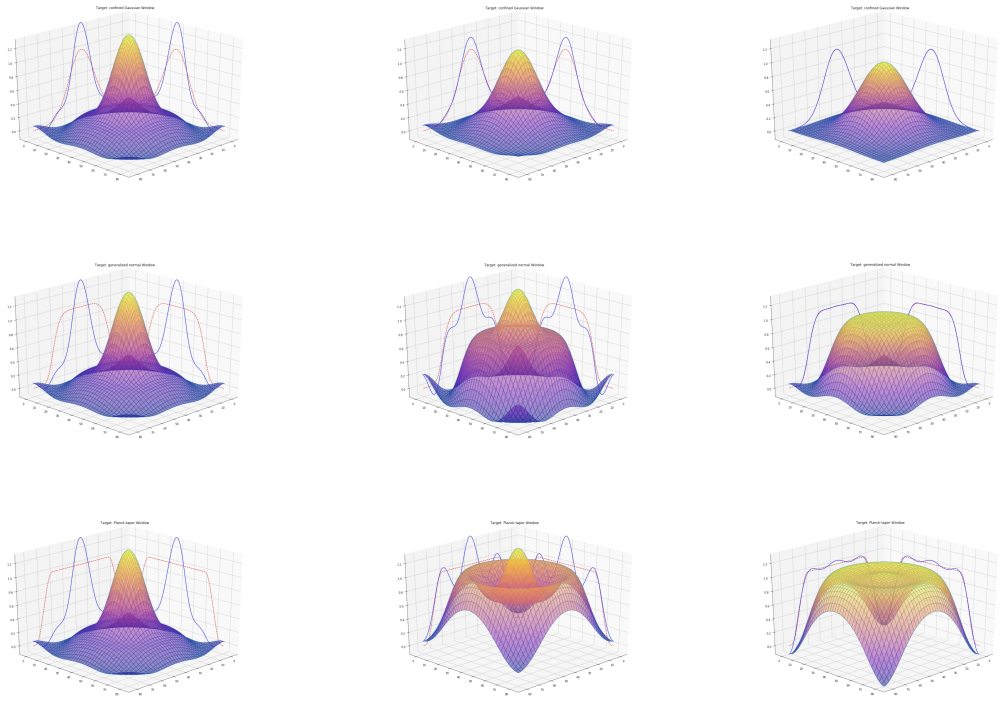

Figure 4.6: Several window functions approximated by a flat top window. *Rows from top to bottom*: confined Gaussian window, generalized normal window and Planck-taper window. *Within a row, from left to right*: inital flat top window, intermediate interpolation step, final interpolation result. *Projected, respectively in blue and red*: current flat top window and target window cross-sections.

Reference (Fleet et al. [26]). Assuming that a high quality estimate of local shifts is essential to generating realistic predictions, the gradient of the prediction errors with respect to the parameters of the window function should allow an automatic focus on observed pixel velocities. To demonstrate the core idea, we use a flat top window (Heinzel et al. [33]) radialized from the weighted sum of cosines

$$w_{FT}[n] = a_0 + \sum_{k=1}^{m} a_k \cos(k\frac{2\pi n}{N}) \tag{4.5}$$

with $m = 4$.

The parameters a can be adjusted by following the respective gradient of a loss function. In a demonstration of the flexibility of the flat top window, it is fitted to a generalized normal window (Chakraborty et al. [39]), a confined Gaussian window (Starosielec et al. [18]) and a Planck-taper window (McKechan et al. [40]). Figure

4.6 demonstrates the process of following the MSE gradient for this purpose. We initialize a to resemble a confined Gaussian window and augment the refinement process by this training option, a is then considered a trainable parameter vector of the preprocessor. While the window function parameters change slightly during training, we have not observed any beneficial nor detrimental effects due to this, but believe these guided adjustments will be necessary, hopefully even interpretable, at higher resolutions.

After this chapter, we are now in the position to present our results and compare them to other existing methods of video prediction.

5. Results

To evaluate how well video prediction via local phase differences with refinement applied in advance to the prediction stage performs, we show the quality of predicted frames on several datasets of increasing difficulty. As baselines, we compare this to images predicted by implementations of VLN and PGP provided by Azizi et al. [41], as well as two versions of the FDTN, one using fully connected (FC-FDTN), and one using convolutional (Conv-FDTN) refinement layers, both implemented by Farazi et al. [9]. Relative performance is quantified by measurements of binary cross entropy loss (BCE), L1 loss, MSE and DSSIM averaged over the validation sets of the respective datasets.

5.1. Prediction of Global Translation

We begin by training all models on *lin_1*, a collection of 5000 sequences of temporal length 10 showing an individual MNIST digit moving on a white background without changes to its orientation nor scale. In this task, the basic assumptions of the motion models hold for each reviewed architecture, so it can provide a meaningful baseline.

Metric	BCE	L1	MSE	DSSIM	no. Params
VLN	3.35e−2	5.42e−3	8.05e−4	6.60e−3	1320429
PGP	3.67e−2	8.49e−3	1.78e−3	1.61e−2	32061
Conv-FDTN	3.27e−2	8.15e−4	2.91e−5	1.22e−4	51065
FC-FDTN	3.14e−2	5.39e−4	1.25e−5	2.46e−4	410108
Ours	3.26e−2	1.25e−3	4.85e−5	5.02e−4	714194

Table 5.1: Errors on the *lin_1* dataset.

An overview of the results is provided in Table 5.1. Despite the additional burden of forming the global prediction from local predictions in 9×9 windows viewed at a hop size of 2, our model outperforms VLN and PGP, and offers comparable prediction quality to both FC-FDTN and Conv-FDTN.

Overall, we believe that the Conv-FDTN presents the most convincing results, especially given that it uses comparatively few parameters. To provide a quali-

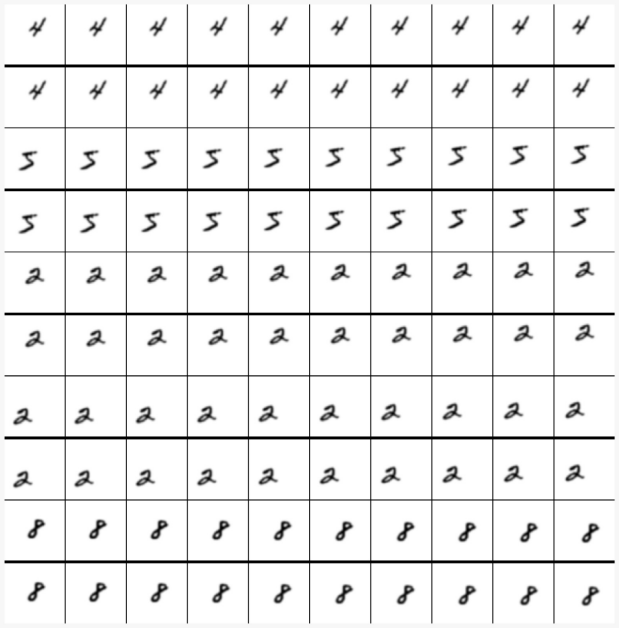

Figure 5.1: The output of the Conv-FDTN on *lin_1*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

tative comparison, we show the predictions for a subset of the validation set for Conv-FDTN in Figure 5.1 and for our model in Figure 5.2. As indicated by the low DSSIM loss achieved by both models, all predictions are plausible. Visualizations of the results for the VLN and PGP can be found in Appendix B along with the outputs of the FC-FDTN.

5.2. Prediction of Composite Transformations

Next, we move to a trial on *rot_lin_scale_2*, which contains 3500 sequences of 2 digits moving while rotating and expanding or shrinking. This dataset was generated according to the the manner specified in Section 4.1 and tests specifically for the capabilities that video prediction based on local phase differences is designed to comprehend. Therefore, comparison to the other models, save for VLN, which does not make any assumptions on the frame-to-frame transformations, is not quite fair, but can quantify the importance of representing such movement in the motion model.

An overview of the results is provided in Table 5.2. Our model, with $H = 2$

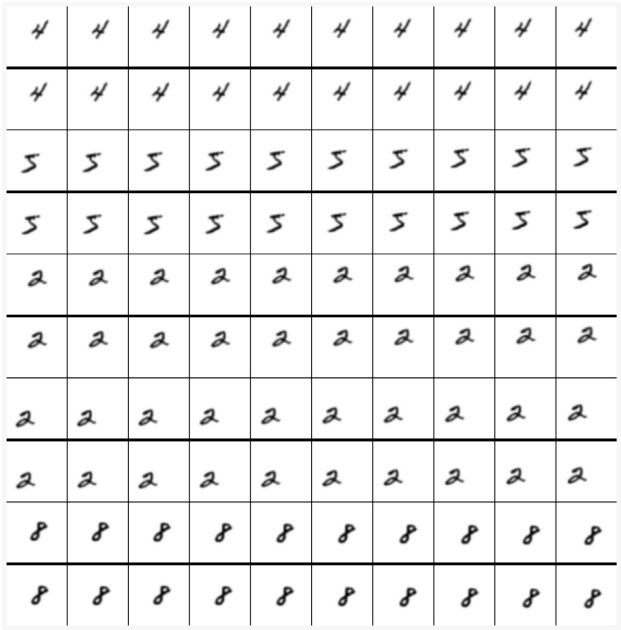

Figure 5.2: The output of our model on *lin_1*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

and a window size of 9, as previously on *lin_1*, significantly outperforms the other methods in this task.

Notably, the VLN performs better in terms of the BCE loss, but this is not critical with respect to visual quality of the generated predictions. This can be observed in Figure 5.3, where the predictions are clearly lacking. For our model, the predicted sequences in Figure 5.4 are more realistic, but the previously mentioned problematic directed drain of image intensity is visible in the bottom row. The images predicted by PGP, FC-FDTN and Conv-FDTN are included in Appendix C.

Metric	BCE	L1	MSE	DSSIM	no. Params
VLN	7.11e−2	1.47e−2	3.41e−3	2.62e−2	1320429
PGP	7.75e−2	2.22e−2	5.61e−3	4.97e−2	32061
Conv-FFT	2.11e−1	4.45e−2	8.61e−3	2.29e−1	51065
FC-FFT	2.08e−1	4.64e−2	8.68e−3	2.34e−1	410108
Ours	7.57e−2	6.30e−3	7.33e−4	6.86e−3	714194

Table 5.2: Errors on the *rot_lin_scale_2* dataset.

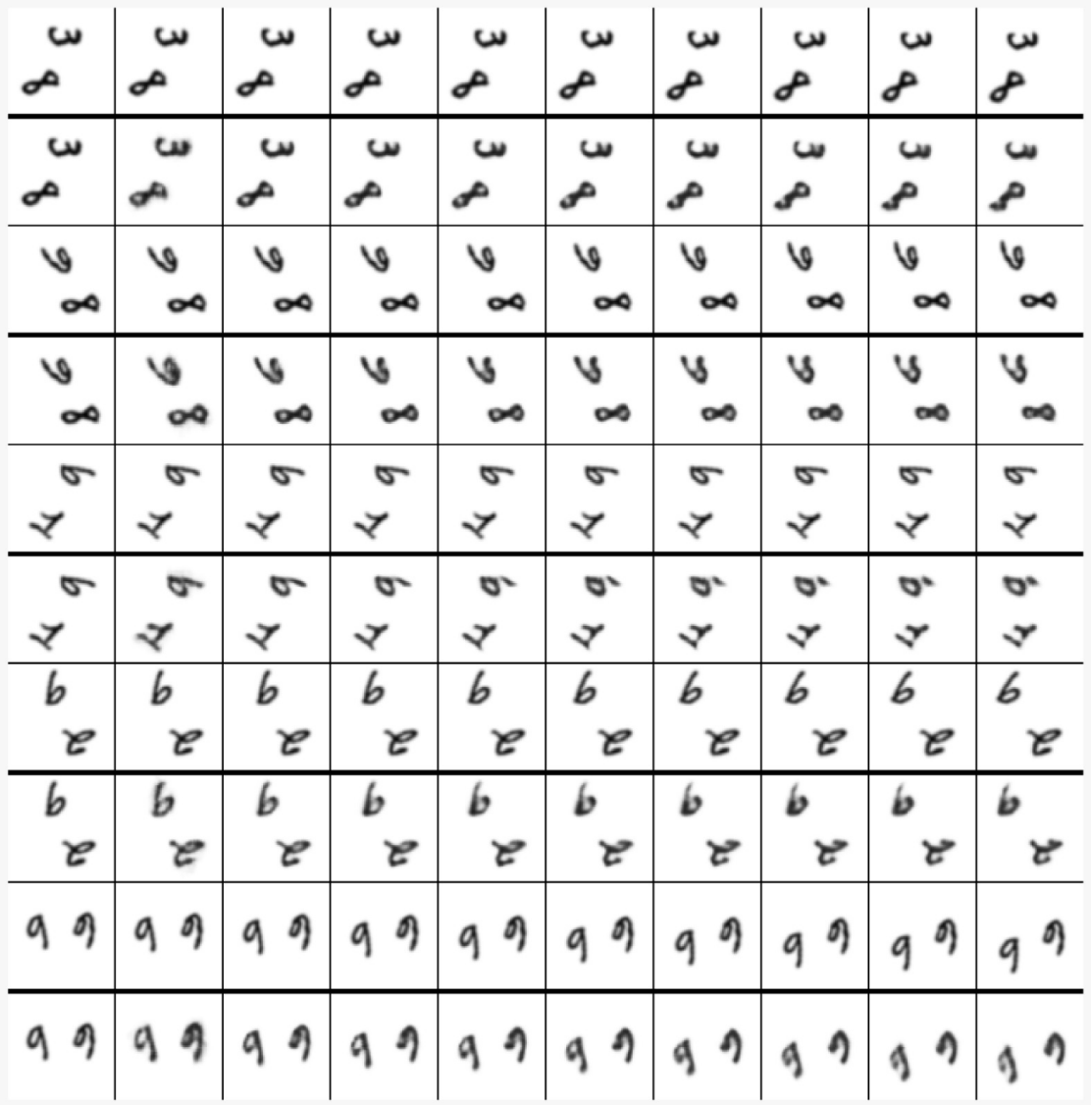

Figure 5.3: The output of the VLN on *rot_lin_scale_2*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

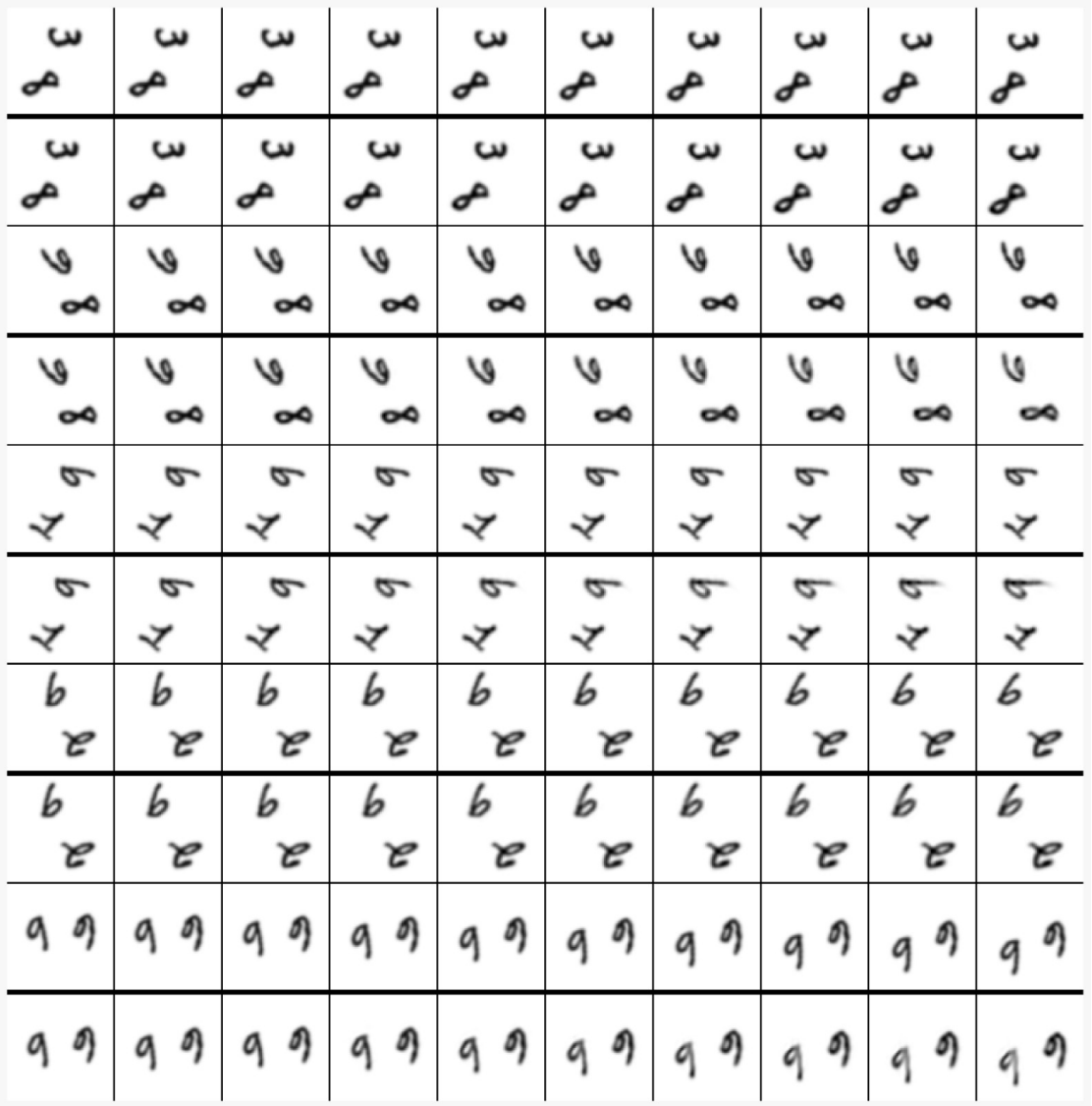

Figure 5.4: The output of our model on *rot_lin_scale_2*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

5.3. Video Prediction on Natural Images

In conclusion to this chapter, we present some preliminary results on natural video sequences from the validation set of sequences we extracted from different traffic camera recordings gathered by Koller et al. [42].

Metric	BCE	L1	MSE	DSSIM
	6.28e−1	6.33e−3	4.16e−4	9.99e−3

Table 5.3: Errors on natural video data

The dataset has about 600 frames in total. Therefore, it is too small to enable training towards very good predictions. However, the validation set errors shown in Table 5.3 are similar to the resulting losses in the previous section, implying that this type of video data is actually easier to predict, potentially because the changes in object scale are not severe.

Figure 5.5: Realistic prediction of a natural video scene.

Figure 5.6: Another example of plausible predictions for this dataset.

In Figures 5.5 and 5.6, the movement of several vehicles is predicted accurately from only 2 seed frames. This can be identified by comparing the position of the cars relative to the road markings and by the front view of the truck leaving the image region in Figure 5.5.

Figure 5.7 shows the intersection recorded from a similar pose with a different camera, but some blurring occurs locally. This effect is amplified for very fast motorists, as exemplified in Figure 5.8.

Figure 5.7: A prediction sequence showing local blurring.

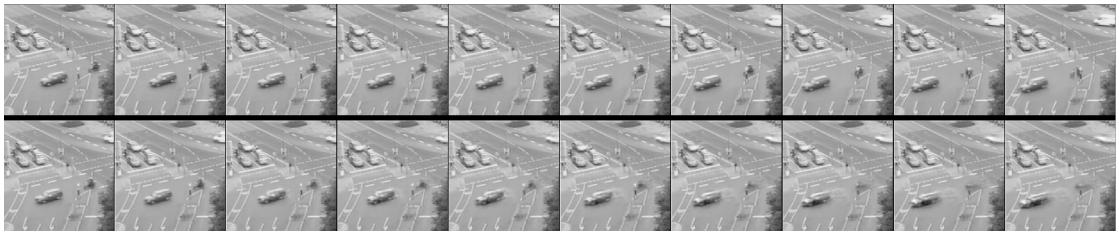

Figure 5.8: In this prediction, significant blurring of the fast car in the center of the scene occurs.

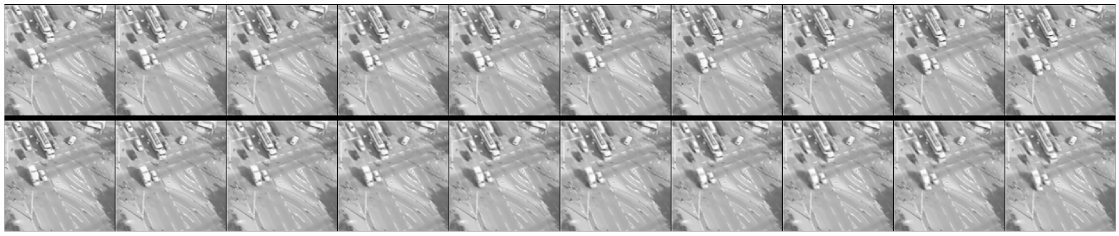

Figure 5.9: Elongated vehicles in predictions of an intersection viewed from above.

Finally, Figure 5.9 shows plausible predictions from a bird's eye view, but several vehicles appear elongated. It should be noted that for this experiment, the output of the refine layer is concatenated to its subsequent input to provide a band-aid solution to the significant accelerations present in these traffic scenes at intersections. In the long term, this method is not acceptable, as it does not allow interpretation of the local accelerations.

In summary, we have shown in this section a first application of our method to traffic data. While the predictions are clearly not perfect in all situations, the results contain some plausible futures for a variety of movements.

6. Conclusion and Outlook

Throughout this thesis, we introduce a video prediction system based on the evaluation and manipulation of local Fourier domain phase information. In order to do so, we first compile the requirements for such a technique, taking into account the advantages and demerits of related work. In this context, we reason that the flexibility to represent arbitrary image transformations, along with their higher order terms, should be prioritized along with end-to-end trainability and exhaustive interpretability of all intermediate results.

In the second chapter, we formalize these ideas, specifying in detail the extraction of local cells and the considerations necessary when making use of the corresponding frequency space representations. In doing so, we have defined the extraction of local phase differences in terms of the local Fourier transform. In addition, we address the visualization of this intermediate result and provide an accompanying sanity check of the validity of local phase differences.

Next, we dedicate a chapter to use of local phase differences as an image transformation. In this context, the assumption of geotemporal permanence emerges as a limit of all local transformation models. Most importantly, we derive the formation of a global frame prediction as the superposition of individual local predictions. Concurrently, we define the inversion rule of the local Fourier transform. Especially in this chapter, theoretical deliberation is inextricably tied to the technical implementation, which proved far more challenging than initially assumed, but is ultimately solved efficiently.

Having previously illustrated a host of mathematical assumptions that we cannot take for granted in practice, we develop a minimalist testbed for various types of image transformations. We combine this with a discussion of adequate loss functions to judge prediction quality and use both tools to weigh multiple refinement approaches, arguing that directly improving the predicted frame carries the risk of undesirable usage of image content knowledge. Therefore, we target the extracted local phase differences as the most promising option for trained improvements to the quality of image forecasting.

Finally, we provide qualitative and quantitative comparison to other video prediction methods on select datasets, supporting the viability of our strategy and thereby rounding out this thesis.

Unfortunately, our end result is yet somewhat unsatisfying. Our video prediction, while evidentially capable of competing with established methods, has thus far not produced convincing results on natural video sequences over a significant time horizon. This shortcoming is compounded with the current lack of explicit representation of higher order terms of the image transformation, which turned out to be beyond the scope of this work.

Nevertheless, we are far from discrediting the local phase-based approach to modeling transformations. Alternative convolutional or otherwise sparse refinement processes less restrictive with respect to utilizing larger local windows are conceivable, potentially even in frequency space. Porting the existing code to TorchScript could accelerate the outer loop of successive frame prediction to the speed of a compiled programming language, thereby rendering quick evaluation on larger datasets possible. Perhaps most excitingly, this could enable a 'prediction pyramid', operating simultaneously at different resolutions and fine-tuned for distinct local velocities at each level and similar to biological systems observed by Baba et al. [43].

At minimum, we hope that we could shed some light on the vast quantity of information enmeshed in video data and its potential to be unraveled by following prediction errors. It is there that we see the main application of video prediction; a key that can unlock semi-supervised learning of proxy tasks on image sequences. We are curious for future developments in this domain and hope to participate in them!

A. Derivation of the Overlap-Equations

The following is a step-by step derivation of the 2D overlap-add equations presented in Chapter 3. Note that the 2D case is entirely analogous to the 1D case.

$$
\begin{aligned}
&\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}\tilde{x}_{t+1,u,v}[n,m]w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}(\frac{1}{N^2}\sum_{\omega_1}^{N-1}\sum_{\omega_2}^{N-1}\mathcal{X}_{t+1,u,v}[\omega_1,\omega_2]e^{j\frac{2\pi}{N}(\omega_1(n-uH)+\omega_2(m-vH))})w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}(\frac{1}{N^2}\sum_{\omega_1}^{N-1}\sum_{\omega_2}^{N-1}(\sum_{k=uH}^{uH+N-1}\sum_{l=vH}^{vH+N-1}x_{t+1,u,v}[k,l]e^{-j\frac{2\pi}{N}(\omega_1(k-uH)+\omega_2(l-vH))})e^{j\frac{2\pi}{N}(\omega_1(n-uH)+\omega_2(m-vH))})w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}(\frac{1}{N^2}\sum_{\omega_1}^{N-1}\sum_{\omega_2}^{N-1}\sum_{k=uH}^{uH+N-1}\sum_{l=vH}^{vH+N-1}x_{t+1,u,v}[k,l]e^{-j\frac{2\pi}{N}(\omega_1(k-n)+\omega_2(l-m))})w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}\sum_{k=uH}^{uH+N-1}\sum_{l=vH}^{vH+N-1}x_{t+1,u,v}[k,l](\frac{1}{N^2}\sum_{\omega_1}^{N-1}\sum_{\omega_2}^{N-1}e^{-j\frac{2\pi}{N}(\omega_1(k-n)+\omega_2(l-m))})w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}\sum_{k=uH}^{uH+N-1}\sum_{l=vH}^{vH+N-1}x_{t+1,u,v}[k,l]\delta[k-n,l-m]w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}x_{t+1,u,v}[n,m]w^a[n-uH,m-vH] \\
&=\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}x_{t+1}[n,m]w[n-uH,m-vH]w^a[n-uH,m-vH] \\
&=x_{t+1}[n,m]\sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty}w^{a+1}[n-uH,m-vH].
\end{aligned}
$$

This proves Equation 3.7 in Chapter 3.

B. Results for Prediction of Global Translation

Below, we present the outputs of the PGP (Figure B.1), the VLN (Figure B.2) and the FC-FDTN (Figure B.3) on the *lin_1* dataset. FC-FDTN produces correct predictions, but VLN and PGP do not accomplish this.

Figure B.1: The output of the PGP on *lin_1*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 3 ground truth seed frames and 7 subsequent predicted frames.

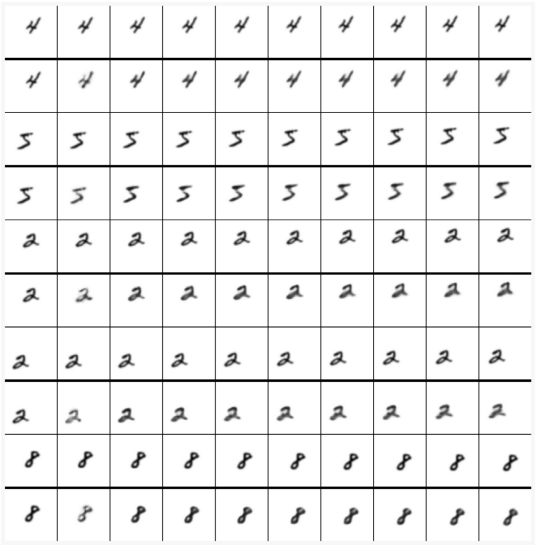

Figure B.2: The output of the VLN on *lin_1. In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

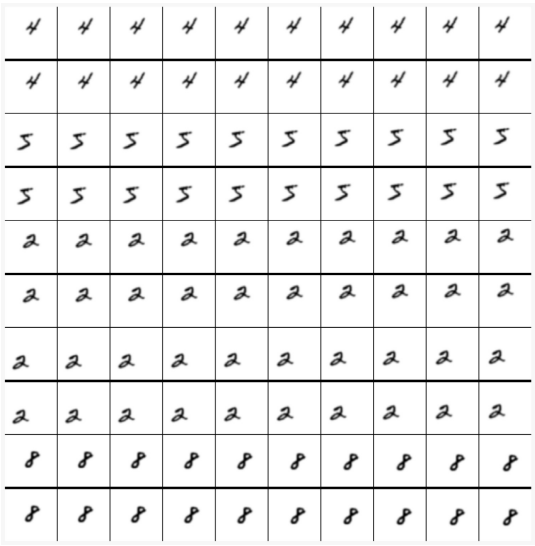

Figure B.3: The output of the FC-FDTN on *lin_1. In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

C. Results for Prediction of Composite Transformations

Below, we present the outputs of the PGP (Figure C.1), the Conv-FDTN (Figure C.2) and the FC-FDTN (Figure C.3) on the *rot_lin_scale_2* dataset. All models presented here, by design of their motion models, cannot account for the transformations present in these frames. Due to this, the predictions quickly diverge from the ground truth data.

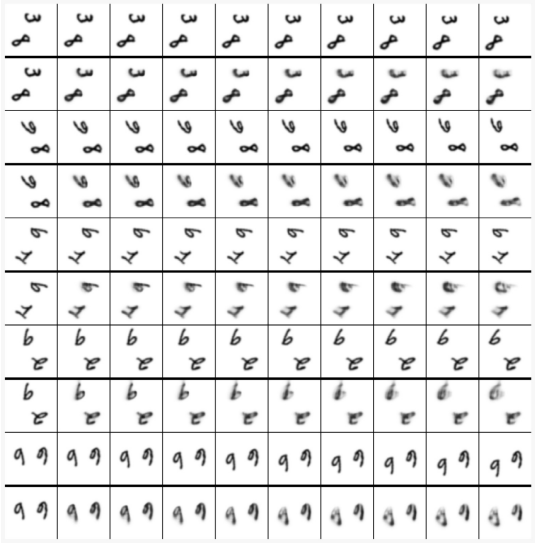

Figure C.1: The output of the PGP on *rot_lin_scale_2*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 3 ground truth seed frames and 7 subsequent predicted frames.

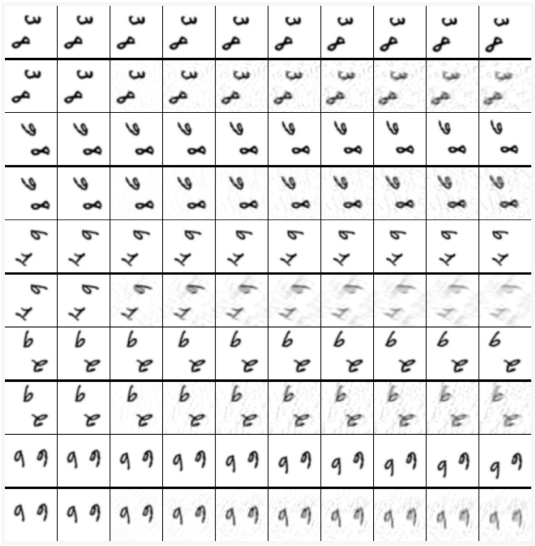

Figure C.2: The output of the Conv-FDTN on *rot_lin_scale_2*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

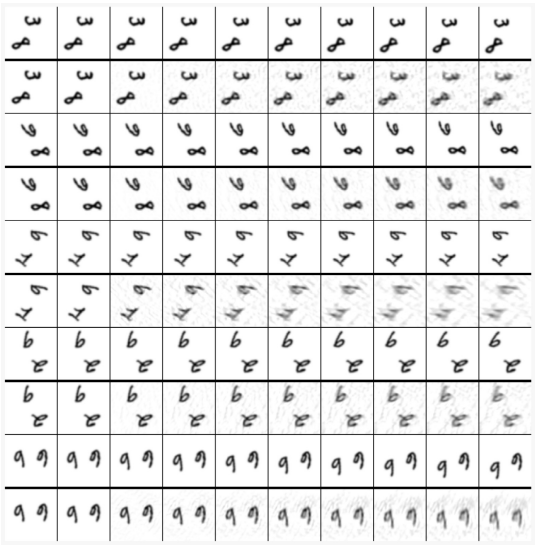

Figure C.3: The output of the FC-FDTN on *rot_lin_scale_2*. *In pairs of two rows*:
Upper row: Ground truth image sequence, arranged in chronological order from left to right.
Lower row, separated by a bold horizontal line: 2 ground truth seed frames and 8 subsequent predicted frames.

List of Figures

List of Tables

Bibliography

[1] J. Stone. *Sesame Street - Mr. Hooper bakes Oscar a baked bean sundae.* Episode 0056. 1970.

[2] J. E. van Engelen and H. H. Hoos. "A survey on semi-supervised learning". In: *Machine learning* 109.2 (2020), pp. 373–440.

[3] W. Lotter, G. Kreiman, and D. D. Cox. "Deep predictive coding networks for video prediction and unsupervised learning". In: *Corr* abs/1605.08104 (2016). arXiv: 1605.08104. URL: http://arxiv.org/abs/1605.08104.

[4] F. Cricri, X. Ni, M. Honkala, E. Aksu, and M. Gabbouj. "Video ladder networks". In: *Corr* abs/1612.01756 (2016). arXiv: 1612.01756. URL: http://arxiv.org/abs/1612.01756.

[5] N. Srivastava, E. Mansimov, and R. Salakhutdinov. "Unsupervised learning of video representations using lstms". In: *Corr* abs/1502.04681 (2015). arXiv: 1502.04681. URL: http://arxiv.org/abs/1502.04681.

[6] V. Michalski, R. Memisevic, and K. Konda. "Modeling deep temporal dependencies with recurrent grammar cells"
". In: *Advances in neural information processing systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 1925–1933. URL: http://papers.nips.cc/paper/5549-modeling-deep-temporal-dependencies-with-recurrent-grammar-cells.pdf.

[7] J. R. van Amersfoort, A. Kannan, M. Ranzato, A. Szlam, D. Tran, and S. Chintala. "Transformation-based models of video sequences". In: *Corr* abs/1701.08435 (2017). arXiv: 1701.08435. URL: http://arxiv.org/abs/1701.08435.

[8] H. Jiang, E. G. Learned-Miller, G. Larsson, M. Maire, and G. Shakhnarovich. "Self-supervised depth learning for urban scene understanding". In: *Corr* abs/1712.04850 (2017). arXiv: 1712.04850. URL: http://arxiv.org/abs/1712.04850.

[9] H. Farazi and S. Behnke. "Frequency domain transformer networks for video prediction". In: *Corr* abs/1903.00271 (2019). arXiv: 1903.00271. URL: http://arxiv.org/abs/1903.00271.

[10] H. Farazi and S. Behnke. "Motion segmentation using frequency domain transformer networks". In: *Arxiv preprint arxiv:2004.08638* (2020).

[11] K. Takita, T. AOKI, Y. SASAKI, T. HIGUCHI, and K. KOBAYASHI. "High-accuracy subpixel image registration based on phase-only correlation". In: *Ieice transactions on fundamentals of electronics, communications and computer sciences* E86-A (Aug. 2003).

[12] B. Reddy and B. Chatterji. "An fft-based technique for translation, rotation, and scale-invariant image registration". In: *Ieee transactions on image processing : a publication of the ieee signal processing society* 5.8 (1996), pp. 1266–1271. ISSN: 1057-7149. URL: https://doi.org/10.1109/83.506761.

[13] J. W. Cooley and J. W. Tukey. "An algorithm for the machine calculation of complex fourier series". In: *Mathematics of computation* 19 (1965), pp. 297–301.

[14] B. Sharpe. *Invertibility of overlap-add processing*. https://gauss256.github.io/blog/cola.html. Online Essay - accessed 20.04.20. 2018.

[15] X. Li. "A simplified normalization operation for perfect reconstruction from a modified stft". In: *2014 12th international conference on signal processing (icsp)*. 2014, pp. 42–45.

[16] R. N. Bracewell. *The fourier transform and its applications /*. 3rd ed. McGraw-Hill, 2000.

[17] Feng Zhou, Ju Fu Feng, and Qing Yun Shi. "Texture feature based on local fourier transform". In: *Proceedings 2001 international conference on image processing (cat. no.01ch37205)*. Vol. 2. 2001, 610–613 vol.2.

[18] S. Starosielec and D. Hägele. "Discrete-time windows with minimal rms bandwidth for given rms temporal width". In: *Signal processing* 102 (Sept. 2014), pp. 240–246.

[19] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Second Edition. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[20] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[21] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. "A database and evaluation methodology for optical flow". In: *International journal of computer vision* 92.1 (2011), pp. 1–31.

[22] Y. Wu, J. Lim, and M.-H. Yang. "Object tracking benchmark". In: *Ieee transactions on pattern analysis and machine intelligence* 37 (Sept. 2015), pp. 1–1.

[23] A. Reyes, A. Alba, and E. Arce-Santana. "Optical flow estimation using phase only-correlation". In: vol. 7. May 2013.

[24] K. Fleischer, M. Haag, F. Heimes, and S. Noltemeier. *Recording of Karl-Wilhelm-Straße - Freiburg*. `http://i21www.ira.uka.de/image_sequences/`. Published Online - accessed 29.06.20. 1997.

[25] D. J. Fleet and Y. Weiss. "Optical flow estimation". In: *Handbook of mathematical models in computer vision*. 2006.

[26] D. J. Fleet and A. D. Jepson. "Computation of component image velocity from local phase information". In: *International journal of computer vision* 5.1 (1990), pp. 77–104.

[27] Y. LeCun, C. Cortes, and C. Burges. "Mnist handwritten digit database". In: *Att labs [online]. available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[28] J. Allen. "Short term spectral analysis, synthesis, and modification by discrete fourier transform". In: *Ieee transactions on acoustics, speech, and signal processing* 25.3 (1977), pp. 235–238.

[29] D. Griffin and Jae Lim. "Signal estimation from modified short-time fourier transform". In: *Ieee transactions on acoustics, speech, and signal processing* 32.2 (1984), pp. 236–243.

[30] A. Röbel. *Analysis/resynthesis with the short time fourier transform*. Aug. 2006.

[31] Scipy. *scipy.signal.chec_COLA*. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.check_COLA.html`. Online Code Documentation - accessed 27.06.20. 2001.

[32] J. O. Smith. *Spectral audio signal processing*. online book, 2011 edition - accessed 10.04.20. `http://ccrma.stanford.edu/~jos/sasp/`, 2011.

[33] G. Heinzel, A. Rüdiger, and R. Schilling. "Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new at-top windows". eng. In: (2002). URL: `http://edoc.mpg.de/display.epl?mode=doc&id=395068`.

[34] P. Svoboda, M. Hradis, D. Barina, and P. Zemcík. "Compression artifacts removal using convolutional neural networks". In: *Corr* abs/1605.00366 (2016). arXiv: `1605.00366`. URL: `http://arxiv.org/abs/1605.00366`.

[35] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *Ieee transactions on image processing* 13.4 (2004), pp. 600–612.

[36] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. R. Bradski. "Kornia: an open source differentiable computer vision library for pytorch". In: *2020 ieee winter conference on applications of computer vision (wacv)* (2020), pp. 3663–3672.

[37] E. Shelhamer, J. Long, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Corr* abs/1605.06211 (2016). arXiv: 1605.06211. URL: http://arxiv.org/abs/1605.06211.

[38] M. Mozer. "A focused backpropagation algorithm for temporal pattern recognition". In: *Complex systems* 3 (Jan. 1995).

[39] D. Chakraborty and N. Kovvali. "Generalized normal window for digital signal processing". In: *2013 ieee international conference on acoustics, speech and signal processing*. 2013, pp. 6083–6087.

[40] D. J. A. McKechan, C. Robinson, and B. S. Sathyaprakash. "A tapering window for time-domain templates and simulated signals in the detection of gravitational waves from coalescing compact binaries". In: *Classical and quantum gravity* 27 (2010), p. 084020.

[41] N. Azizi, H. Farazi, and S. Behnke. "Location dependency in video prediction". In: *Icann*. 2018.

[42] D. Koller, H. Kollnig, K. Fleischer, M. Haag, F. Heimes, S. Noltemeier, G. Eichberger, H. Leuck, and H.-H. Nagel. *Recording of from various traffic cameras*. http://i21www.ira.uka.de/image_sequences/. Published Online by KOGS/IAKS Universität Karlsruhe - accessed 29.06.20. 1992-1998.

[43] M. Baba, K. S. Sasaki, and I. Ohzawa. "Integration of multiple spatial frequency channels in disparity-sensitive neurons in the primary visual cortex". In: *Journal of neuroscience* 35.27 (2015), pp. 10025–10038. ISSN: 0270-6474. eprint: https://www.jneurosci.org/content/35/27/10025.full.pdf. URL: https://www.jneurosci.org/content/35/27/10025.