




2018

Learning, Moving, And Predicting With Global Motion Representations

Andrew Coulter Jaegle

University of Pennsylvania, drewjaegle@gmail.com

Follow this and additional works at: <https://repository.upenn.edu/edissertations>

 Part of the [Artificial Intelligence and Robotics Commons](#), and the [Neuroscience and Neurobiology Commons](#)

Recommended Citation

Jaegle, Andrew Coulter, "Learning, Moving, And Predicting With Global Motion Representations" (2018). *Publicly Accessible Penn Dissertations*. 2877.

<https://repository.upenn.edu/edissertations/2877>

This paper is posted at ScholarlyCommons. <https://repository.upenn.edu/edissertations/2877>

For more information, please contact repository@pobox.upenn.edu.

Learning, Moving, And Predicting With Global Motion Representations

Abstract

In order to effectively respond to and influence the world they inhabit, animals and other intelligent agents must understand and predict the state of the world and its dynamics. An agent that can characterize how the world moves is better equipped to engage it. Current methods of motion computation rely on local representations of motion (such as optical flow) or simple, rigid global representations (such as camera motion). These methods are useful, but they are difficult to estimate reliably and limited in their applicability to real-world settings, where agents frequently must reason about complex, highly nonrigid motion over long time horizons. In this dissertation, I present methods developed with the goal of building more flexible and powerful notions of motion needed by agents facing the challenges of a dynamic, nonrigid world. This work is organized around a view of motion as a global phenomenon that is not adequately addressed by local or low-level descriptions, but that is best understood when analyzed at the level of whole images and scenes. I develop methods to: (i) robustly estimate camera motion from noisy optical flow estimates by exploiting the global, statistical relationship between the optical flow field and camera motion under projective geometry; (ii) learn representations of visual motion directly from unlabeled image sequences using learning rules derived from a formulation of image transformation in terms of its group properties; (iii) predict future frames of a video by learning a joint representation of the instantaneous state of the visual world and its motion, using a view of motion as transformations of world state. I situate this work in the broader context of ongoing computational and biological investigations into the problem of estimating motion for intelligent perception and action.

Degree Type

Dissertation

Degree Name

Doctor of Philosophy (PhD)

Graduate Group

Neuroscience

First Advisor

Kostas Daniilidis

Keywords

Computational neuroscience, Computer vision, Deep learning, Machine learning, Motion, Vision

Subject Categories

Artificial Intelligence and Robotics | Computer Sciences | Neuroscience and Neurobiology

LEARNING, MOVING, AND PREDICTING WITH GLOBAL MOTION REPRESENTATIONS

Andrew Jaegle

A DISSERTATION

in

Neuroscience

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

2018

Supervisor of Dissertation

Kostas Daniilidis, Ruth Yalom Stone Professor of Computer and Information Science

Graduate Group Chairperson

Joshua I. Gold, Professor of Neuroscience

Dissertation Committee:

Johannes Burge, Assistant Professor of Psychology (Chair)

Diego Contreras, Professor of Neuroscience

Nicole Rust, Associate Professor of Psychology

Jianbo Shi, Professor of Computer and Information Science

Katerina Fragkiadaki, Assistant Professor of Machine Learning, CMU

For Alice

ACKNOWLEDGEMENT

I have been privileged to have the support of a fantastic group of colleagues in neuroscience and computer science during my doctoral studies. I would like to thank my advisor, Kostas Daniilidis, for his mentorship and encouragement. His intuitions and rigor have been crucial to the development of the ideas in this dissertation. I would also like to thank Diego Contreras who provided mentorship and scientific guidance, especially in my first few years at Penn.

I am grateful to the members of both the Daniilidis and Contreras labs. Among many others, Morgan Taylor, Madineh Sarvestani, Iván Fernández de Lamo, Stephen Phillips, Daphne Ippolito, Oleh Rybkin, and Karl Pertsch were the source of countless discussions and inspiration. Thanks to Ken Chaney and Nikos Kolotouros for computing support. Special thanks to Kosta Derpanis for consistently providing structure for our discussions and work.

Thanks to the members of my committee, Johannes Burge, Nicole Rust, Jianbo Shi, and Katerina Fragkiadaki for their advice and encouragement. Thanks to David Brainard and the faculty and staff associated with the IGERT Complex Scene Perception training grant for creating a space of genuine interdisciplinary focus. I'm grateful to the members of the Neuroscience Graduate Group, and for the collaborative and supportive atmosphere the students, faculty, and staff have built.

I had the privilege to spend summers at the Max Planck Institute for Intelligent Systems in Tübingen and at DeepMind in London. I am grateful to my collaborators there, and especially to my hosts: Michael Black and Javier Romero at the MPI and Greg Wayne at DeepMind. I am grateful to them for their intellectual generosity.

I am incredibly grateful to my friends and family for providing my life with the structure and support needed to follow this work through. Thanks to Dave Barack for many fruitful discussions at the interface of philosophy, neuroscience, and artificial intelligence. Thanks to my bandmate, John McMillin, for providing me with an outlet for musical creativity. Thanks to my parents and siblings for their love and support. And most of all, thanks to my partner, Alice Xie, who has always

challenged me to be my best.

This work would not have been possible without the invaluable contributions of collaborators. The work presented in Chapter 2 was done jointly with Stephen Phillips and Kostas Daniilidis; the work presented in Chapter 3 was done jointly with Stephen Phillips, Daphne Ippolito, and Kostas Daniilidis; and the work presented in Chapter 4 was done jointly with Oleh Rybkin, Kosta Derpanis, and Kostas Daniilidis.

ABSTRACT

LEARNING, MOVING, AND PREDICTING WITH GLOBAL MOTION REPRESENTATIONS

Andrew Jaegle

Kostas Daniilidis

In order to effectively respond to and influence the world they inhabit, animals and other intelligent agents must understand and predict the state of the world and its dynamics. An agent that can characterize how the world *moves* is better equipped to engage it. Current methods of motion computation rely on local representations of motion (such as optical flow) or simple, rigid global representations (such as camera motion). These methods are useful, but they are difficult to estimate reliably and limited in their applicability to real-world settings, where agents frequently must reason about complex, highly nonrigid motion over long time horizons. In this dissertation, I present methods developed with the goal of building more flexible and powerful notions of motion needed by agents facing the challenges of a dynamic, nonrigid world. This work is organized around a view of motion as a global phenomenon that is not adequately addressed by local or low-level descriptions, but that is best understood when analyzed at the level of whole images and scenes. I develop methods to: (i) robustly estimate camera motion from noisy optical flow estimates by exploiting the global, statistical relationship between the optical flow field and camera motion under projective geometry; (ii) learn representations of visual motion directly from unlabeled image sequences using learning rules derived from a formulation of image transformation in terms of its group properties; (iii) predict future frames of a video by learning a joint representation of the instantaneous state of the visual world and its motion, using a view of motion as transformations of world state. I situate this work in the broader context of ongoing computational and biological investigations into the problem of estimating motion for intelligent perception and action.

ACKNOWLEDGEMENT	iv
ABSTRACT	vi
LIST OF TABLES	x
LIST OF ILLUSTRATIONS	xviii
CHAPTER 1 : Introduction	1
1.1 Overview	1
1.2 Visual motion analysis: local, global, and beyond	2
1.2.1 From local to global motion	3
1.2.2 Motion, representation, and spatiotemporal invariances	7
1.2.3 Designing and learning dynamic representations	10
1.2.4 Motion and prediction	12
1.3 Summary of contents	13
CHAPTER 2 : Fast, robust, continuous monocular egomotion computation	16
2.1 Introduction	16
2.2 Related work	19
2.2.1 Egomotion/visual odometry	19
2.2.2 Continuous, monocular approaches	19
2.2.3 Robust optimization	21
2.3 Problem formulation and approach	22
2.3.1 Visual egomotion computation and the motion field	22
2.3.2 Robust formulation	25
2.3.3 Confidence weight estimation by expected residual likelihood	26

2.3.4	Robust estimation using a lifted kernel	29
2.4	Experiments	31
2.4.1	Evaluation on KITTI	32
2.4.2	Synthetic sequences	34
2.5	Conclusions	35
CHAPTER 3 : Understanding image motion with group representations		37
3.1	Introduction	37
3.2	Related work	39
3.2.1	Motion representations	39
3.2.2	Learning representations using visual structure	40
3.3	Approach	41
3.3.1	Group properties of motion	41
3.3.2	Learning motion by group properties	42
3.3.3	Sequence learning with neural networks	44
3.4	Experiments	45
3.4.1	Rigid motion in 2D	46
3.4.2	Real-world motion in 3D	47
3.5	Conclusion	51
CHAPTER 4 : Predicting the future with transformational states		52
4.1	Introduction	52
4.2	Related work	54
4.3	Approach	57
4.3.1	Transformational states	59
4.3.2	Weighted residual connections	61
4.4	Experiments	63
4.4.1	Datasets	63
4.4.2	Architecture and training details	65

4.4.3	Evaluation	67
4.5	Conclusion	69
CHAPTER 5: Conclusion		71
5.1	Future work	71
5.1.1	Developing functional DNN models of the dorsal stream	71
5.1.2	Long-term prediction for control	73
5.1.3	Discovering degrees of freedom	74
5.1.4	Learning symmetries	75
5.2	Summary	77
APPENDICES		79
A	Supplemental material: Fast, robust, continuous monocular egomotion computation	79
A.1	Supplemental experiments	79
A.2	Derivation of linear least squares estimate	82
A.3	Lifted weights formulation	84
A.4	Implementation details of Soatto/Brockett algorithm	84
B	Supplemental material: Understanding image motion with group representations	90
B.1	Additional Experiments	90
C	Supplemental material: Predicting the future with transformational states	94
C.1	Video results	94
C.2	Network architectures	94
C.3	Comparison to other prediction models	99
C.4	Ablation studies	99
BIBLIOGRAPHY		102

List of Tables

TABLE 1 :	Average embedding error (equation 3.1) on held-out data. Results are averaged over forward, backward, and loop sequences. Errors are relative to a chance error of 1: values lower than 1 indicate that equivalent (inequivalent) motions are close together (far apart) in the embedding space.	44
TABLE 2 :	Linear regression from the learned embedding to the translation and rotation of the KITTI odometry dataset consistently performs better than chance (guessing the mean value). Table entries show mean squared error \pm standard error (percent improvement).	47
TABLE 3 :	Interpolation distances on KITTI (as in Figure 11), averaged across test data. Distances are consistently lower for the true frame than for visually similar frames (inside sequence) and dissimilar frames (outside sequence) when using the embedding, but not the Euclidean distance.	48
TABLE 4 :	Comparison of binary cross-entropy (BCE) results (nats/frame) on the Moving MNIST test set. Lower scores indicate better performance. . . .	63
TABLE 5 :	Comparison of frame prediction results on the KTH test set. Higher scores indicate better performance.	64
TABLE 6 :	Comparison of next frame prediction results on the UCF101 test set (split 1). Higher scores indicate better performance.	64
TABLE 7 :	Comparison of sequence prediction model components and training configurations.	100

List of Figures

- FIGURE 1 : Schematic depiction of the ERL method for egomotion estimation from noisy flow fields. Figure best viewed in color. (A) Example optical flow field from two frames of KITTI odometry (sequence 5, images 2358-2359). Note the outliers on the grass in the lower right part of the image and scattered throughout the flow field. (B) We evaluate the flow field under M models with translation parameters sampled uniformly over the unit hemisphere. The residuals for the flow field under three counterfactual models are shown. Each black point indicates the translation direction used. Residuals are scaled to $[0,1]$ for visualization. (C) We estimate the likelihood of each observed residual under each of the models by fitting a Laplacian distribution to each set of residuals. The final confidence weight for each flow vector is estimated as the expected value of the residual likelihood over the set of counterfactual models. Likelihood distributions are shown for the three models above. (D) The weighted flow field is used to make a final estimate of the true egomotion parameters. The black point indicates the translation direction estimated using ERL and the green point indicates ground truth. The unweighted estimate of translation is not visible as it is outside of the image bounds. 17
- FIGURE 2 : A 2D line-fitting problem demonstrating how ERL weights inliers and outliers. Inliers are generated as $y_i \approx 2x_i + 1$ with Gaussian noise. Each data points is colored according to its estimated confidence weight. . . . 27

FIGURE 3 :	Robust methods recover the error surface of the outlier-free flow field. (A) Example optical flow field from two frames of KITTI odometry (sequence 10, images 14-15). Note the prominent outliers indicated by the yellow box. Error surfaces on this flow field for (A) the raw method (equation (2.9)) with all flow vectors, (B) with outliers removed by hand, and (C) with confidence weights estimated by ERL or (D) the lifted kernel. The green point is the true translational velocity and the black point the method's estimate. Blue: low error. Red: high error. Translation components are given in calibrated coordinates.	28
FIGURE 4 :	Median translational and rotational errors on the full KITTI odometry dataset for our methods and baselines.	30
FIGURE 5 :	Full distribution of translational velocity errors.	33
FIGURE 6 :	Full distribution of rotational velocity errors.	33
FIGURE 7 :	Translation error as a function of percent outliers on synthetic data for our robust methods and two baseline continuous egomotion methods.	35
FIGURE 8 :	(a) A graphical model describing the relationship between the latent scene structure $\{\mathcal{S}_t\}$, motion $\{\mathcal{M}_t\}$, and the observed images of a sequence. We describe a method for learning a representation $\overline{\mathcal{M}}$ of the motion space \mathcal{M} from observed image sequences $\{I_t\}$. (b) By recomposing sequences of images to satisfy the group properties of associativity and invertibility, we construct pairs of image sequences with equivalent motion. We use these properties to learn an approximate group homomorphism $\Phi \in \overline{\mathcal{M}}$ between motion in the world and in an embedding.	39

FIGURE 9 : **(a)** Network structure. The RNN output at the final step of the sequence is treated as the sequence embedding. During training, the distance between sequence embeddings is adjusted using an embedding loss. **(b)** We recompose sequences to enforce associativity and invertibility. Sequences with equivalent motion (e.g. 1-2-4 and 1-3-4) serve as positive examples, while sequences with inequivalent motions (e.g. 1-2-4 and 4-3-1) serve as negative examples. 41

FIGURE 10 : **(a)** An example test sequence from MNIST and the corresponding saliency maps. Saliencies show the gradient backpropagated from the final RNN timestep. Each column represents an image pair passed to one of the CNNs. **(b)-(d)** tSNE of the network embedding on the test set, with points labeled by **(b)** the magnitude of translation in pixels, **(c)** the translation direction in degrees, and **(d)** the digit label (0-9). The representation clusters sequences by both translation magnitude and direction but not identity. 46

FIGURE 11 :	(a) Natural image motion is a subspace of the space of all image transformations, and a particular motion can be viewed as a path in the latent space of natural images. Although $[I_1, I_K, I_T]$ has a total transformation equivalent to $[I_1, I_T]$ for any value of K , only $[I_1, I_m, I_T]$ can be produced by natural image motion. (b) In interpolation experiments, we compare the distance between the embedding of $[I_1, I_T]$ with the embedding of this sequence after inserting either a true middle frame (I_m) or another frame (I_{IN} or I_{OUT}). (c) Images with lowest relative error taken from the sequence or from the whole dataset, for each distance measure. Errors are relative to that of the true middle frame in the corresponding measure: high relative errors ($\gg 1$) indicate the distance distinguishes realistic motion from unrealistic motion. Images other than true middle frame produce dramatically higher errors when using the embedding but not when using a Euclidean distance.	49
FIGURE 12 :	Saliency results on a test sequence from KITTI tracking with both camera and independent motion. The network focuses on areas that are relevant to determining motion in 3D, not simply regions with large temporal image gradients.	50

FIGURE 13 : (A) This work is motivated by the observation that image transformations may be more easily modeled by a network that learns to transform latent states rather than transform or associate pixel intensities. By learning to model states, s , along with transformations between states, g , the RNN is encouraged to model sequences not by memorizing arbitrary transitions between certain images (static embedding) but by reshaping the embedding so that natural state transformations are predictable (transformational embedding). Figure best viewed in color. (B) Sample predictions of our model on sequences from the Moving MNIST and KTH datasets. We show only the last of 10 input images for visualization purposes. Our model produces good image predictions using only pixel-wise reconstruction losses. 53

FIGURE 14 : Architecture overview. (A) Our model uses an encoder-decoder sequence-to-sequence architecture with a factorized latent that captures the image state, s , and transformation, d . Residual connections are omitted for clarity; see the text and Figure 15 for details. (B) Future states are transformed from past states using an RNN core that accumulates the transformation estimate g with a ConvLSTM and applies it to the recursively estimated state s with an operator CNN_{Φ} 58

FIGURE 15 :	Weighted residual connections. (A) To produce high quality images at multiple time steps in the future without re-encoding images, we use a residual connection scheme designed to gradually alter image content from the last observed input image. Residual connections connect the encoder at time T (last input) to the decoder at time $T + 1$ (first output). At subsequent times, the decoder inherits information about the past from the decoder at the previous time only. The network has this connectivity pattern at every layer: we show only two decoder layers and the output image for easier visualization. (B) We use a retinotopic weighting scheme to allow each layer of a decoding network to selectively incorporate skipped input from the past. Weights and feature maps at time t are functions of the predicted latent state \hat{s}_t at time t	60
FIGURE 16 :	Example sequences on Moving MNIST. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. Our model is able to stably predict digits over multiple timesteps, even when digits overlap for multiple frames.	61
FIGURE 17 :	Example sequences on KTH. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. The model produces faithful motion in a variety of settings and is able to paint in the background after dis-occlusion.	65
FIGURE 18 :	Example sequences on UCF101. For each example, we show two frames from the past followed by the ground truth third frame and the third frame predicted by the model from the first two images.	66

FIGURE 19 :	Example failure cases on KTH. (a) The model outputs a blurry motion sequence that does not correspond to the ground truth. (b) The model fails to correctly predict motion or paint in the background when the moving object occupies only a small part of the image. (c) The model fails to correctly paint in the background after the foreground moves, leading to ghosting artifacts.	69
FIGURE 20 :	Log likelihoods of Laplacian and Gaussian fits to the optical flow error. Laplacian fits are consistently better than Gaussian fits.	80
FIGURE 21 :	Median translational and rotational errors on the full KITTI odometry dataset for ERL with two candidate distributions.	81
FIGURE 22 :	Error on egomotion regression from self-supervised flow PCA as a function of the number of principal components included. Horizontal lines reflect our method (latent, shown in red) and a chance baseline (shown in green).	91
FIGURE 23 :	Cumulative percent variance explained of the optical flow in KITTI odometry as a function of the number of principal components included. 67% of the variance is explained by the first 5 components; 90% of the variance is explained by the first 40 principal components.	92
FIGURE 24 :	Representative principal components of optical flow on the KITTI odometry dataset. The first few components capture the dominant motions (forward and left/right turning) and reflect the stereotypical depth structure of KITTI.	93

FIGURE 25 :	Comparison of Moving MNIST results on architectures with and without residual connections. The model labeled “no residuals” has no skip or residual connections of any kind. The model labeled “w/ residuals” is the full model described in the paper. The model without weighted residual connections produces good predictions, but including these connections produces crisper results, especially at early prediction time steps. Both architectures reliably capture digit identity, even after the digits overlap.	100
FIGURE 26 :	Comparison of KTH results on models with architectural ablations. (i) “No transformational state”: the RNN core omits the CNN_Φ and includes only ConvLSTM components. (ii) “Residuals skipped from last input image”: each decoder directly receives residual input from the encoder at the last input time step ($t = T$) instead of the previous decoder time step. Weighted residuals are still used. (iii) “No residuals”: no residual or skip connections of any kind are used. The second sequence shown here is very challenging for all models. The full model produces better motion (notice the motion of the legs) and less prominent ghosting artifacts than ablations.	101
FIGURE 27 :	Additional comparisons of KTH results on models with architectural ablations. See Figure 26 caption for explanation of ablations.	102

CHAPTER 1 : Introduction

1.1. Overview

The visual world is complex and ever changing, and its complexity extends from local structures to more extended patterns in space and time. For an intelligent agent to effectively engage the world, it must learn to reason about how the state of the world is changing, and how local states and changes reflect the larger patterns and structures of the world.

Visual motion is at the core of these issues, and in this dissertation we develop methods towards the goal of understanding motion. For an agent to make best use of motion, it is important that its representations do not merely reflect the low-level statistics of the world: they should reflect the full set of properties observable by an agent. Accordingly, I focus on the problem of understanding global motion. By global motion, I mean the motion of the full image and ultimately the scene, not just of individual pixels. Understanding motion in this sense will facilitate the development of methods enabling intelligent agents to use the changes observed in the world to understand it and to act in it.

The body of this dissertation is composed of three works, which develop methods for motion analysis and motion-based prediction. Before presenting the technical contributions of this dissertation, I survey the state of motion analysis by reviewing the computational problem of motion analysis, local approaches to motion and their relationship to global methods, and challenges to the current state of motion research motivated by the computational goals of prediction and spatiotemporally invariant representation. In chapters 2, 3, and 4, I present work developed with collaborators that addresses the limitations of our understanding of the computation and use of motion described above. I conclude the dissertation by describing promising future avenues of research suggested by the work presented here.

In this chapter, I describe the broader context of motion analysis in computer vision, neuroscience, and artificial intelligence and situate the presented work in this context. In particular, I describe (i) the limitations of local and simple global representations of motion, (ii) how deep neural networks

(DNNs) can be used to learn more flexible global representations by allowing learning in more general motion frameworks and by tying representation learning to tasks, and (iii) how motion and visual prediction can be used to develop spatiotemporally invariant representations.

1.2. Visual motion analysis: local, global, and beyond

By visual motion, I mean changes over time of the intensity of light in an image (formed by a camera or a camera-like retina) due to changes in the state of the world. In the settings discussed here, I will limit the discussion to image sequences produced by contiguous motion: that is, by the motion induced as the world changes in time around an observer, which may itself moves through the world smoothly¹. These changes may be due to the motion of the camera, to the motion of objects in the scene, or some combination thereof.

One goal of visual motion analysis is to determine what source of motion in the world led to motion in the image and to characterize the patterns in this motion. This view leads naturally to local analyses of motion, which address the proximal cause of motion by estimating the motion of pixels between pairs of frames. In some settings, such as egomotion estimation, local motion can be used to estimate motion at the level of the full image. Egomotion estimation uses the optical flow field to estimate camera rotation and translation (up to a scale). These and other methods can be used to estimate what motion is due to the camera and what to other objects, to track objects in time, and to reason about the position of the observer in a static map of its environment.

Fundamentally, these methods are concerned with making correspondences between the world at subsequent time steps. The goal of making correspondences is just one part of a broader computational challenge in motion: using motion to understand and make inferences about the structure of the world not just as it is, but as it changes in time, and how to use these inferences to act in the dynamic world in a dynamic manner. In this dissertation, I will describe tools and methods directed at solving this problem.

¹I thus avoid any discussion of changes in image sequences produced by cuts, saccades, or other discontinuities. Understanding image transformations in these domains present another set of challenges, and they are an important avenue for future work.

In the next sections, I summarize the state of motion analysis and the current challenges, working from low-level to high-level. First, I describe local motion estimation and how it can be used to extract certain global representations of image motion. The relationship between local and global estimation is further developed in Chapter 2, where I describe a method for robust estimation of egomotion from optical flow. Second, I describe the limitations of global motion, and how these limitations can be used to motivate a view of representations more directly tied to the goals of general perception. One approach to learning such a representation is developed in Chapter 3, where I describe a method for unsupervised learning of motion representations. Third, I describe a computational view of motion representations in terms of the efficient development of spatiotemporal invariants. I use this to motivate a discussion of the relationship between motion and prediction. Fourth, I describe how prediction can be used as a task to learn representations of motion and efficiently capture spatiotemporal invariants of the scene. This material is further developed in Chapter 4, where I describe how motion can be used in a model for visual prediction to produce naturalistic image predictions.

1.2.1. From local to global motion

It is informative to first examine how local motion is typically computed, and how this local motion is brought into relationship with the global motion of an image. Classical approaches to motion typically analyze motion in terms of spatiotemporally local changes. Perhaps the prototypical example of this form of analysis is optical flow (Gibson 1950; Horn and Schunck 1981). Optical flow attempts to find correspondences between pixels in I_t , the image at time t , and pixels in I_{t+1} . Many methods have been proposed to solve this (e.g. Lucas and Kanade 1981; Black and Anandan 1993; Farneback 2003; Sun et al. 2010; Brox and Malik 2011; Revaud et al. 2015), and indeed recent years have seen a number of advances in this area (e.g. Fischer et al. 2015; Sevilla-Lara et al. 2016; Ranjan and Black 2017).

Optical flow and other local representations of motion have been hypothesized to be computed in the retina and early visual system. In primates, there is strong evidence that representations of visual motion are computed in the dorsal cortical pathway beginning in primary visual cortex (V1) and

continuing to medial temporal (MT) and medial superior temporal (MST) areas, as well as later areas such as the ventral interparietal area (VIP). The outputs of these areas project to areas involved in decision making (e.g. the lateral interparietal area (LIP)) and guiding motor control (e.g. the caudal interparietal area (CIP)), as well as to areas such as those in the superior temporal sulcus (STS), where they are integrated with projections from ventral areas encoding the static content of the scene (Singer and Sheinberg 2010). The early stages of this pathway contain populations of neurons highly sensitive to the direction of motion of visual stimuli (Jones and Palmer 1987; Rust et al. 2005). Motion sensitive neurons in V1 project preferentially to MT (Movshon and Newsome 1996), where neurons exhibit sensitivity to local motion invariant to other properties of the stimulus (Dubner and Zeki 1971). Neurons in later areas are selective for stimuli with more complex dependence on motion, such as egomotion (Gu et al. 2010), object motion (Mineault et al. 2012), action recognition (Giese and Poggio 2003), motion-based motor planning (Grefkes and Fink 2005), and decision making (Newsome et al. 1989; Gold and Shadlen 2000; Shadlen and Newsome 2001). The emerging view is that the dorsal stream computation progressively computes representations of 3D visual motion (Sanada and DeAngelis 2014; Sunkara et al. 2016; Sasaki et al. 2017; Chen et al. 2018).

Models of local motion estimation in neural systems are exemplified by two approaches: local energy filters (Adelson and Bergen 1985) and Reichardt detectors (Hassenstein and Reichardt 1956; van Santen and Sperling 1985). Local energy filters are designed to detect the presence of energy along a plane in the spatiotemporal spectral domain, while Reichardt detectors are simple delay-line filters designed to respond when a simple pattern of light moves from one position to another. These two models are designed to detect motion at a specific velocity, and both have been used as functional models for motion sensitive neurons. These two frameworks are equivalent in some settings (Adelson and Bergen 1985), and I will focus the discussion here on energy filters, which are more widely used in the literature on primate motion processing².

Models built using energy filter-like computations can be used to obtain good fits to the steady-state responses of visual cortex neurons implicated in motion processing (Simoncelli and Heeger 1998;

²In contrast, Reichardt detectors and related models are widely used as models and putative explanations of motion processing in the fly nervous system (Borst et al. 2010).

Rust et al. 2006) when combined with other biologically motivated components, such as divisive normalization (Carandini and Heeger 2011). Energy filters can be used to build estimates of optic flow with the introduction of mechanisms designed to overcome the local ambiguity known as the aperture problem³. This ambiguity arises in 2D motion estimation at small windows because, as the window of estimation becomes small, the spatial structure becomes effectively 1D and the problem of estimating 2D motion becomes under-constrained. For more detailed descriptions of this ambiguity and computational perspectives on how to resolve it, see (Horn 1986 Chapter 12, Pack et al. 2003).

Local motion can be used to estimate global motion by introducing simplifying assumptions about the nature of the motion being observed. In egomotion (or camera motion), the world is assumed to be rigid (i.e. its motion is given by a single rotation and translation) and the motion given by the motion of the camera or observer. An analysis of this situation in terms of its projective geometry gives a simple relationship between the translation and rotation of the camera, the depth at each observed point, and the optical flow field (the motion field equation, given in equation (2.1)). Given optical flow, perhaps in the presence of Gaussian noise, this equation can be solved to estimate the camera translation and rotation, even without knowledge of the depth structure of the scene (Heeger and Jepson 1992).

Egomotion estimation is quite fragile in real world situations because it relies on optical flow estimation, which is often unreliable and noisy, and because it assumes the world is rigid, which is often violated in real-world settings due to the presence of independently moving objects. Nonetheless, stable and reliable egomotion estimation is essential for current navigation and control algorithms. We present a method for making camera motion estimation more robust by addressing these two sources of noise in chapter 2⁴. Subsequent methods have used deep neural networks to arrive at better estimates of egomotion and related quantities (Costante et al. 2016; Zhou et al. 2017; Fragkiadaki et al. 2017; Costante and Ciarfuglia 2017).

³The mechanisms for achieving flow-like computation in the brain are not sufficient to obtain reasonable performance on state-of-the-art optical flow benchmarks (Solari et al. 2014; Chessa et al. 2015). This gap in performance is wide enough to suggest that computing optical flow, without qualification, is not an adequate description of the computational role of these areas in the brain.

⁴We also discuss the related problems of structure from motion (SfM) (Tomasi and Kanade 1992) and simultaneous localization and mapping (SLAM) (Scaramuzza and Fraundorfer 2011) in chapter 2.

Several neural populations have responses that have been described or modeled as coding for egomotion or object motion⁵. Because of the well-known geometrical and computational relationship between egomotion and flow, several groups have proposed models of a dorsal area one synapse downstream from MT in terms of egomotion or object motion computation (Perrone and Stone 1994; Grossberg et al. 1999; Hamed et al. 2003; Browning 2012; Mineault et al. 2012). This area, MST, contains some cells that are driven preferentially by local optical flow patterns of rotation, translation, divergence, convergence, and other deformations (Tanaka et al. 1986; Graziano et al. 1994; Duffy and Wurtz 1995), which resemble patterns predicted by models of egomotion estimation from flow (Heeger and Jepson 1992; Perrone and Stone 1998; Zemel and Sejnowski 1998). MST neurons project to areas with putative roles in grasp formation (Zhang and Sejnowski 1999), eye movement (Pack et al. 2001; Takemura et al. 2007), and action recognition (Giese and Poggio 2003; Singer and Sheinberg 2010).

We can thus see that the standard view of rigid local-to-global motion analysis in computer vision (from pixels to flow to camera and object motion) has an analogue in the functional properties of early visual cortical processing (from V1 to MT to MST)⁶. Models that test this relationship typically do so by recording the activity of the neurons in question while presenting stimuli with known optical flow or egomotion. This analysis is only possible because it is possible to regress the responses of neurons in these areas to known quantities (optical flow, camera motion, and object motion) that are known and can be easily manipulated. .

This strategy is not generally feasible for motion stimuli more complex than simple camera or object motion. Visual motion-based methods for solving action recognition, visual prediction, and other higher-level perceptual and control problems are not typically phrased in terms of known, analytical relationships between motion representations. Instead, they typically involve optimization and learning. This makes it difficult to directly compare the methods used by a high performing

⁵From an algorithmic point of view, purely visual egomotion and object-motion estimation are identical problems if the motion of the scene is rigid (i.e. given in terms of a rotation and translation), up to a change in coordinates.

⁶A similar story emerges in dorsal areas of the cat visual system (areas 17 and 18 and the lateral suprasylvian (LS) and anterior ectosylvian visual (AEV) areas) (Palmer et al. 1978; Toyama and Kozasa 1982; Gizzi et al. 1990; Scannell et al. 1996; Li et al. 2000; Ouellette et al. 2004). The existence of this functional homology emphasizes the importance of strategies for perception that rely on visual motion.

artificial system with functionally homologous areas in the brain. The problem is exacerbated by the high dimensionality associated with the complex spatiotemporal patterns arising in contexts beyond simple 2D motion. The curse of dimensionality for large, temporally extended motions makes it infeasible to design parameterized stimuli and experimental protocol to adequately probe the stimulus space, and hence very difficult to plausibly establish that specific computations are instantiated in a given neuron or area ⁷. In the next section, I turn to the question of designing representations that can form the basis for better models of motion analysis and perhaps allow for better understanding of the full computation of the dorsal stream.

1.2.2. Motion, representation, and spatiotemporal invariances

As we have seen, egomotion and related methods make restrictive assumptions about the world: typically, that it is rigid or that it contains only object of a known class (as in nonrigid structure from motion (Bregler et al. 2000)). Because of this, these methods are limited in their applicability to more general motion analysis, and they cannot be easily made more general, as the methods are designed to exploit these simplifying assumptions. The strategy at the core of these methods is to (i) estimate a known metric quantity (such as optical flow), (ii) analyze how it is related to another metric quantity, such as camera motion, and (iii) estimate the second quantity using estimates of the first. If this strategy could be extended to compute more complex metric quantities (e.g. to the set of motions characterizing human locomotion), would it be adequate?

This strategy is adequate if we are content to view the computational modules of a motion analysis system as black boxes, each of whose input-output mapping is configured separately, and whose internal computations are inaccessible to other modules. A setup like this is common in systems for simultaneous localization and mapping (SLAM), for example, where optical flow is computed using one algorithm, used to estimate camera pose with another algorithm, and then integrated into a global map using a pipeline relying on these poses (Scaramuzza and Fraundorfer 2011). A black-box systems view is unsatisfactory from our point of view for several reasons: (i) the neural areas

⁷The “hypermotion” stimuli used in Mineault et al. 2012 to characterize neural responses in area MST are close to the limit of what is feasible in practical experiments using parameterized motion stimuli.

instantiating these computations are highly interconnected and exhibit feedback relationships that are likely to be functionally important (Lamme et al. 1998); (ii) the computational models instantiated by these black boxes can be improved by incorporating interaction between modules (Hariharan et al. 2015; Newell et al. 2016; Huang et al. 2017; Fragkiadaki et al. 2017); (iii) end-to-end learning (e.g. by backpropagation) relies on the ability of later results to drive changes in the computational processes used to produce earlier, intermediate results (Rumelhart et al. 1986). All three of these properties are exploited in the methods developed in this dissertation, and they are not easily accommodated by the feed-forward black box view.

Ultimately, the computational goal of motion analysis should be situated in terms of the ethological demands of behavior and the representations used to guide it, rather than on the computation of intermediate representations (Churchland et al. 1994). How do the perceptual components of motion processing, which have so far been the focus of our discussion, relate to this broader picture?

To address this question, It is useful to make a distinction between representations of motion that are *explicit* and those that are *latent*. The kinds of representations discussed so far are explicit representations: they are the result of motion analysis framed in terms of estimates of predefined quantities, such as optical flow or egomotion, which often correspond to known metric properties of the world (the 2D translation of pixels and the 3D translation and rotation of the camera, in the case of flow and egomotion, respectively). When using explicit representations, any subsequent use of the motion analysis that has been performed is solely by means of the explicit representation itself: i.e., the computation used to produce the representation is typically inaccessible to later stages.

In contrast, latent representations of motion are computed as an intermediate step of a model whose final computation has a more general perceptual or task-related target. While implicit representations may correlate with metric targets, this relationship is not typically engineered specifically. This representational strategy is currently exemplified by deep neural network (DNN) models. DNNs are machine learning systems built by composing differentiable modules (layers) and trained to approximate a desired functional mapping (LeCun et al. 2015; Goodfellow et al. 2016). DNNs are designed to take input from a target domain (e.g. images) and to smoothly approximate a mapping

to a target domain by applying a series of elementary differentiable operations (typically additions, multiplications, spatial pooling, and nonlinearities). DNNs are typically trained by comparing this output to a target using a differentiable loss function (such as the cross-entropy loss for classification or the mean-squared error loss for regression) and then minimizing the resulting loss by stochastic gradient descent (SGD) using the backpropagation algorithm (Rumelhart et al. 1986). In so doing, the parameters of the neural network are changed so that full system is able to perform the target task. This results in the development of latent representations in the internal layers of the network.

The output of intermediate layers for DNNs trained in this way on image processing tasks can show selectivity for useful intermediate features (Zeiler and Fergus 2014; Zhou et al. 2015; Olah et al. 2018), and the resulting latent representations have proved to be useful for transfer to other tasks and representations (e.g. Long et al. 2015) and as functional explanations of the activity of brain areas (Yamins and DiCarlo 2016). A DNN trained to perform a task that relies on the perception of dynamic content, such as action recognition, can learn to represent features similar to those seen in explicit representation models without being told about quantities such as optical flow (Karpathy et al. 2014; Varol et al. 2018).

While explicit representations are very useful because they can be analytically manipulated and understood intuitively in terms of correspondences between pixels and objects between time steps⁸, they are fundamentally limited. Rather than extending existing explicit representations, I focus here on developing better latent representations of motion that do not explicitly target metric flow or egomotion. In particular, I focus on representations structured so as to learn to use motion in service of a more general representation with more task relevance. Latent representations developed this way offer the potential to address the problems with explicit, black-box reasoning as presented above. These representations can be learned directly in terms of a final task and they can be learned jointly and interactively with other representations.

⁸Fortunately, latent representations can be related to more intuitive explicit representations in some settings. Several works have shown that representations trained for tasks depending on motion learn internal representations correlated with explicit and interpretable representations (Karpathy et al. 2014; Zhou et al. 2015; Varol et al. 2018). The ability to decode explicit representations of motion from latent representations of motion remains an important test for the success of unsupervised methods (see Chapter 3 for a discussion). This view also gives us a way to interpret results suggesting that visual cortical neurons encode explicit quantities in the world for which they are not directly trained. But, as pointed

In Chapter 3, we show how one latent representation of motion can be learned in an unsupervised manner by formulating global motion representation in terms of the group properties of motion. Other groups have proposed methods to learn motion by using the constraints associated with more local forms of motion, such as optical flow, to induce a loss for DNN training (Yu et al. 2016). While these approaches have furthered our understanding of how motion can be structured and learned to work in real world settings, the question of how to structure these representations to capture not only motion, but how it relates to dynamic structure, remains challenging. We turn to this question in the next section.

1.2.3. Designing and learning dynamic representations

To understand what form dynamic representations should take, it is useful to contrast them to static representations, which are more studied and better understood. One useful way to do this is by comparing the dorsal pathway ($V1 \rightarrow MT \rightarrow MST$ and beyond) to a canonical computational pathway in the ventral stream of the primate brain ($V1 \rightarrow V2 \rightarrow V4 \rightarrow IT$). While the dorsal stream is typically characterized as computing properties of visual motion or dynamics, the ventral stream is typically characterized in terms of its role in computing properties of static features of the visual world, such as shape, form, and object identity (Goodale and Milner 1992).

Areas of the ventral stream after V1 are selective for spatial patterns building in complexity from textures (Freeman et al. 2013) to shapes and objects (Kobatake and Tanaka 1994) and faces (Tsao et al. 2006). That is, the level of abstraction of the representation increases at later stages in the pathway, and the representations move away from pixels and towards behaviorally relevant objects. Recently, several groups have presented compelling evidence that the responses of neurons in this pathway are well modeled by the internal representations in DNNs trained on object recognition (Yamins and DiCarlo 2016).

The ventral stream and models of object recognition in computer vision can be described in terms of the progressive construction of task-related invariant representations (DiCarlo et al. 2012; Anselmi

out in Morcos et al. 2018, we should be careful not to conclude that the main function of an area is building explicit representations, or indeed that the presence or absence of these representations is causally related to the function of the area.

et al. 2016). An *invariant* representation of some input datum x is a function of that datum, $\Theta(x)$, that does not change if an irrelevant transformation (say $t \in T$, where T is the set of irrelevant, or nuisance, transformations) is applied to the datum: $\Theta(x) = \Theta(tx)$. For example, an invariant representation of a particular individual would be the same if that person were facing left or facing right, as this transformation does not change the person's identity. Invariant representations are ubiquitous in computer vision (e.g. Lowe 2004; Laptev 2005): if such a representation can be discovered, it offers a way to relate images to tasks in a manner that is robust to irrelevant changes in the scene.

A story similar to ventral one can be told about the emergence of invariant representations of dynamic features in progressively more abstract dorsal computations (Giese and Poggio 2003; Jhuang et al. 2007). As in Giese and Poggio 2003, the computational goal of higher-order motion processing is often framed in terms of biological motion or action recognition. It is useful to think of these tasks as exemplars of the larger computational problem of representing the dynamic structure in a scene by observing how it can move.

In section 110 of his “An Essay Towards A New Theory of Vision,” George Berkeley describes a similar view of the usefulness of motion for building representations. Here, he contemplates the problem of how a person, born blind and seeing another's body for the first time, would recognize that the parts of the body should be represented as belonging to the same object:

He would not, for Example, make into one complex Idea, and thereby esteem an Unite all those particular Ideas, which constitute the visible Head or Foot. For there can be no Reason assigned why he should do so, barely upon his seeing a Man stand upright before him: There croud into his Mind the Ideas which compose the visible Man, in company with all the other Ideas of Sight perceiv'd at the same time: But all these Ideas offered at once to his View, he would not distribute into sundry distinct Combinations, till such time as by observing the Motion of the Parts of the Man and other Experiences, he comes to know, which are to be separated, and which to be collected together (Berkeley 1709).

Berkeley argues that the relationship between the motions of the parts is what enables efficient learning of the structure of the body. Motion can play a similar role in guiding representation learning (Agrawal et al. 2015; Pathak et al. 2017). By learning the motion of objects, we can learn to predict what will happen as they change in time. We can also distinguish these from other, irrelevant changes, such as changes in image properties or geometry that do not occur in contiguous time. In the next section, I will discuss how motion and prediction can be used for learning.

1.2.4. *Motion and prediction*

The problem of building invariant visual representations is the problem of associating visual images that contain the same content (Rust and Stocker 2010). One way to learn these sorts of representations is by exploiting the fact that, over time, the content of the image changes slowly, and accordingly representations should also change slowly. This is the basic strategy employed by slow feature analysis (SFA) (Wiskott and Sejnowski 2002), and indeed this method has shown success in the context of learning representations using DNNs in recent years (Wang and Gupta 2015). However, at their core, SFA and similar methods posit only that features change slowly. Because of this, they generally are not flexible enough to accommodate multiple spatial and temporal scales of change, which are typically important for accounting for all of the change in a scene⁹.

Visual prediction offers a more general and flexible way to learn representations by learning to associate visual images. In visual prediction, we want to learn a function $f : \mathbb{R}^{N \times T} \rightarrow \mathbb{R}^{N \times K}$ mapping a past image sequence $\{I_1, \dots, I_T\}$ to a future image sequence $\{I_{T+1}, \dots, I_{T+K}\}$, where each image has N pixels. Such a function, if it can be trained on a dataset, can be said to associate images on that dataset, as it matches past images to future images. For this strategy to be fully general, it must also generalize between images of different objects, to reflect the fact that for example, different dogs under the same setting will tend to move in comparable ways. Fortunately, convolutional neural network (CNN) representations of images have shown very good generalization properties when trained on natural images with SGD and other standard techniques (Krizhevsky et al. 2012; Hinton

⁹For example, see Anandan 1989 for a discussion of how accounting for differences of spatiotemporal scale can dramatically improve the quality of optical flow estimates.

et al. 2012; Yosinski et al. 2014). If we parameterize f with a CNN-based network, we can learn a function by SGD that can predict and generalize between images of different object instances. See Chapter 4 for a more in-depth discussion of the problem of visual prediction and a description of current approaches.

Visual prediction gives us a tool for learning representations that associate images of the same content. To learn such a representation efficiently, changes in state from time to time should be as simple as possible. Visual motion offers a way to gain this efficiency: instead of re-describing the world at each time step, we need only describe the relevant changes ¹⁰. Several works have shown that using local motion can lead to reasonable predictions (Brabandere et al. 2016; Pătrăucean et al. 2016; Vondrick and Torralba 2017), but these methods struggle in the same situations where local representations of motion typically struggle: when objects become occluded or dis-occluded and when motion is poorly described in terms of local translation. These situations occur frequently when making predictions over several time steps, so a more global representation is better suited to such situations. In chapter 4, we show that building a future by incorporating a more abstract, global model of motion, that acts on latent representations instead of pixels, can lead to good predictions.

1.3. Summary of contents

In this section, I have described the challenges facing motion representation. I have described how the work in this dissertation addresses the problems of (i) robustly estimating egomotion, (ii) learning to represent global motion in an unsupervised manner, and (iii) learning a representation of motion to predict future frames. I now turn to describing the technical contributions of this dissertation in detail.

The contents of the rest of the dissertation are as follows:

- In chapter 2, I present methods for reasoning about camera motion using noisy estimates of optical flow. We propose the expected residual likelihood (ERL) method, which robustifies egomotion inference using likelihood distributions of local motion residuals under counterfac-

¹⁰This property is commonly exploited in video compression schemes, such as MPEG (Le Gall 1991).

tual parameters of the motion field equation. We show ERL outperforms baselines, including a novel alternative method using a lifted kernel, while introducing minimal runtime overhead. We demonstrate that by incorporating confidence weights based on the known relationship between local and global motion, we can achieve more robust estimates of camera motion from local estimates. This work was previously published as Jaegle et al. 2016.

- In chapter 3, I present a method for learning image motion directly from unlabeled image sequences using a learning rule designed to enforce the group properties of visual motion. Our learning rule is designed to impose the axiomatic properties of groups on a latent representation learned on a dataset, which encourages the latent space to represent the transformation group in that setting. This learning rule is general in the sense that it makes no assumptions about the motion of the camera, structure of the world, or the class of motions seen: all are learned in an unsupervised manner. We show that a model trained with this method learns representations capturing key properties of motion in 2D and 3D settings using an inductive bias that is less restrictive than those based on local motion effects, such as optical flow. This work was previously published as Jaegle et al. 2018a.
- In chapter 4, I present a method for image sequence prediction using a model that learns to represent current states and transform them to estimate the future. Methods for image sequence prediction typically predict a sequence of instantaneous future states and then use these states to reconstruct future frames. We introduce an architecture that learns to represent the transformation between states in addition to the state, and we generate future frames by recursively transforming past states using the estimated transformation. This configuration leads to predicted future sequences with convincing motion, and qualitative and quantitative results competitive with the state of the art on several datasets. This work was previously published in preprint form as Jaegle et al. 2018b.
- In chapter 5, I summarize the presented work and suggest promising directions for future work. I discuss future applications of the work presented here to developing (i) models of dorsal stream function, (ii) methods for long-term visual prediction and action learning, and

(iii) methods for unsupervised learning of the spatio-temporal structure of visual scenes (its degrees of freedom and symmetries).

CHAPTER 2 : Fast, robust, continuous monocular egomotion computation

2.1. Introduction

Visual odometry in real-world situations has attracted increased attention in the past few years in large part because of its applications in robotics domains such as autonomous driving and unmanned aerial vehicle (UAV) navigation. Stereo odometry and simultaneous localization and mapping (SLAM) methods using recently introduced depth sensors have made dramatic progress on real-world datasets. Significant advances have also been achieved in the case of *monocular* visual odometry when combined with inertial information.

State-of-the-art visual odometry uses either the discrete epipolar constraint to validate feature correspondences and compute inter-frame motion (Scaramuzza and Fraundorfer 2011) or directly estimates 3D motion and 3D map alignment from image intensities (Forster et al. 2014). In contrast to the state of the art, in this paper we revisit the *continuous* formulation of structure from motion (SfM), which computes the translational and rotational velocities and depths up to a scale from optical flow measurements. Our motivation lies in several observations:

- UAV control schemes often need to estimate the translational velocity, which is frequently done using a combination of monocular egomotion computations and single-point depths from sonar (Bristeau et al. 2011).
- Fast UAV maneuvers require an immediate estimate of the direction of translation (the focus of expansion) in order to compute a time-to-collision map.
- Continuous SfM computations result in better estimates when the incoming frame rate is high and the baseline is very small.

However, estimating camera motion and scene parameters from a single camera (*monocular* egomotion estimation) remains a challenging problem. This problem case arises in many contexts where sensor weight and cost are at a premium, as is the case for lightweight UAVs and consumer cameras. Situations involving monocular sensors on small platforms pose additional problems: computational

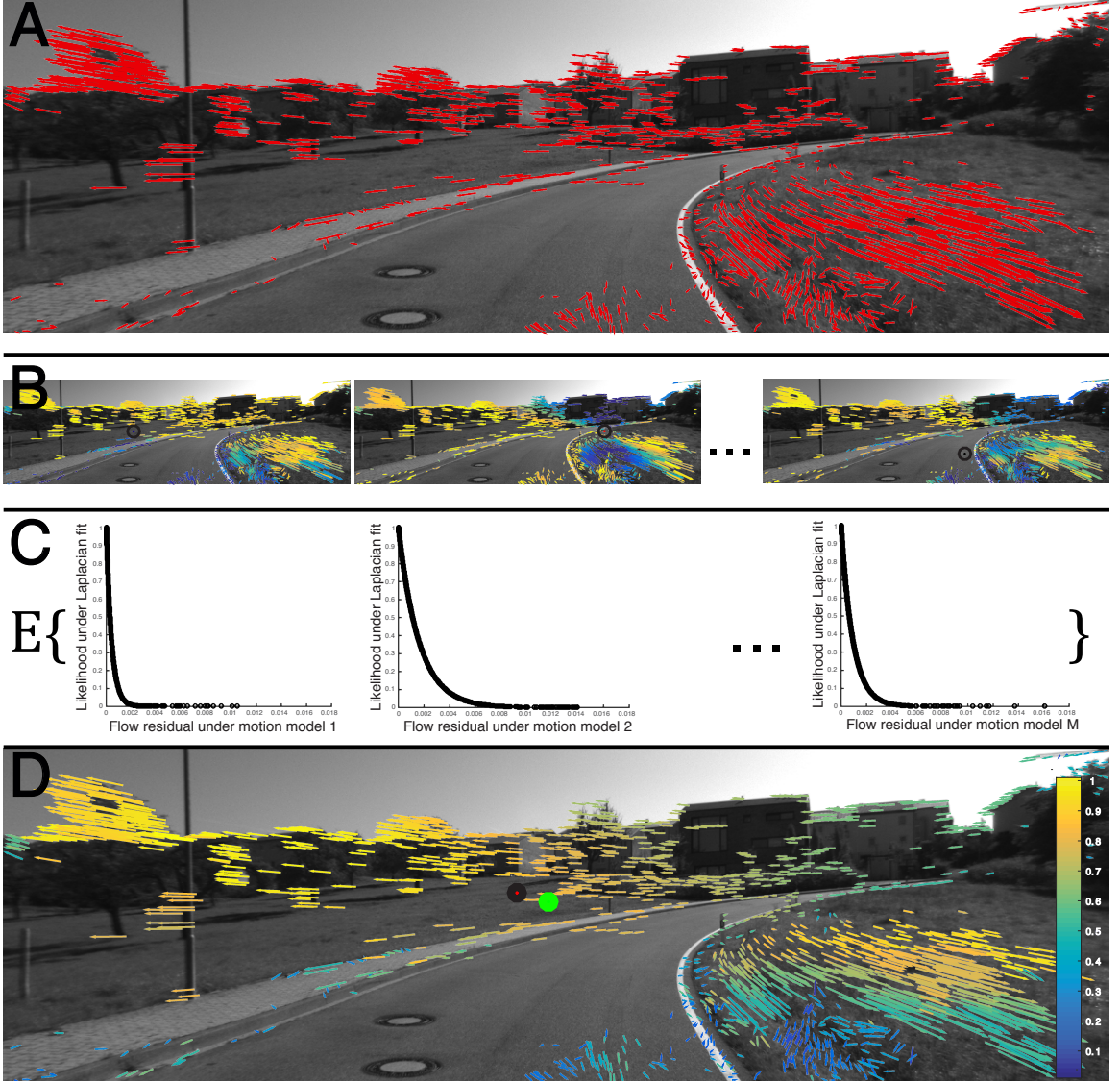


Figure 1: Schematic depiction of the ERL method for egomotion estimation from noisy flow fields. Figure best viewed in color. (A) Example optical flow field from two frames of KITTI odometry (sequence 5, images 2358-2359). Note the outliers on the grass in the lower right part of the image and scattered throughout the flow field. (B) We evaluate the flow field under M models with translation parameters sampled uniformly over the unit hemisphere. The residuals for the flow field under three counterfactual models are shown. Each black point indicates the translation direction used. Residuals are scaled to $[0,1]$ for visualization. (C) We estimate the likelihood of each observed residual under each of the models by fitting a Laplacian distribution to each set of residuals. The final confidence weight for each flow vector is estimated as the expected value of the residual likelihood over the set of counterfactual models. Likelihood distributions are shown for the three models above. (D) The weighted flow field is used to make a final estimate of the true egomotion parameters. The black point indicates the translation direction estimated using ERL and the green point indicates ground truth. The unweighted estimate of translation is not visible as it is outside of the image bounds.

resources are often very limited and estimates must be made in real time under unusual viewing conditions (e.g. with a vertically flipped camera, no visible ground plane, and a single pass through a scene). These contexts present many sources of noise. Real-time flow estimation produces unreliable data, and the associated noise is often pervasive and non-Gaussian, which makes estimation difficult and explicit outlier rejection problematic. Furthermore, violations of the assumption of scene rigidity due to independent motion of objects in the scene can lead to valid flow estimates that are outliers nonetheless. Even in the noise-free case, camera motion estimation is plagued with many suboptimal interpretations (illusions) caused by the hilly structure of the cost function. Additionally, forward motion, which is very common in real-world navigation, is known to be particularly hard for monocular visual odometry (Oliensis 2005).

We propose an algorithm suitable for the robust estimation of camera egomotion and scene depth from noisy flow in real-world settings with high-frame-rate video, large images, and a large number of noisy optical flow estimates. Our method runs in real-time on a single CPU and can estimate camera motion and scene depth in scenes with noisy optical flow with outliers, making it suitable for integration with filters for real-time navigation and for deployment on light-weight UAVs. The technical contributions of this paper are:

- A novel robust estimator based on the expected residual likelihood (ERL) of flow data that effectively attenuates the influence of outlier flow measurements and runs at 30-40 Hz on a single CPU.
- A novel robust optimization strategy using a lifted kernel that modifies the shape of the objective function to enable joint estimation of weights and model parameters, while enabling good empirical convergence properties.

2.2. Related work

2.2.1. *Egomotion/visual odometry*

Many approaches to the problem of visual odometry have been proposed. A distinction is commonly made between feature-based methods, which use a sparse set of matching feature points to compute camera motion, and direct methods, which estimate camera motion directly from intensity gradients in the image sequence. Feature-based approaches can again be roughly divided into two types of methods: those estimating camera motion from point correspondences between two frames (*discrete* approaches) and those estimating camera motion and scene structure from the optical flow measurements induced by the motion between the two frames (*continuous* approaches). In practice, point correspondences and optical flow measurements are often obtained using similar descriptor matching strategies. Nonetheless, the discrete and continuous approaches use different problem formulations, which reflect differing assumptions about the size of the baseline between the two camera positions.

The continuous approach is the appropriate choice in situations where the real-world camera motion is slow relative to the sampling frequency of the camera. Our approach is primarily intended for situations in which this is the case, e.g. UAVs equipped with high-frame-rate cameras. Accordingly, we focus our review on continuous, monocular methods. For a more comprehensive discussion, see Ma et al. 2004.

2.2.2. *Continuous, monocular approaches*

In the absence of noise, image velocities at 5 or 8 points can be used to give a finite number of candidate solutions for camera motion (Longuet-Higgins and Prazdny 1980, Hartley and Zisserman 2003, Nistér 2004). With more velocities, there is a unique optimal solution under typical scene conditions (Horn 1988). Many methods have been proposed to recover this solution, either by motion parallax (Longuet-Higgins and Prazdny 1980; Hildreth 1992; Heeger and Jepson 1992; Jepson and Heeger 1990) or by using the so-called continuous epipolar constraint (Ma et al. 2004). The problem is nonlinear and nonconvex, but various linear approximation methods have been proposed to simplify

and speed up estimation (Jepson and Heeger 1991; Zhuang et al. 1988; Kanatani 1993).

Although the problem has a unique optimum, it is characterized by many local minima, which pose difficulties for linear methods (Chiuso et al. 2000). Furthermore, in the presence of noise, many methods are biased and inconsistent in the sense that they do not produce correct estimates in the limit of an unlimited number of image velocity measurements (Zhang and Tomasi 2002). Many methods also fail under many common viewing conditions or with a limited field of view (Daniilidis and Nagel 1990). Recently, Fredriksson et al. 2014 and Fredriksson et al. 2015 proposed branch-and-bound methods that estimate translational velocity in real time and effectively handle a large numbers of outliers. However, these methods deal with the case of pure translational camera motion, while our approach estimates both translational and rotational motion.

Most directly related to our work is the robust estimation framework presented in Zhang and Tomasi 1999. They propose a method based on a variant of a common algebraic manipulation and show that this manipulation leads to an unbiased, consistent estimator. They pose monocular egomotion as a nonlinear least-squares problem in terms of the translational velocity. In this framework, angular velocity and inverse scene depths are also easily recovered after translational velocity is estimated. To add robustness, they use a loss function with sub-quadratic growth, which they solve by iteratively reweighted least squares (IRLS). We use a similar formulation but demonstrate several novel methods for estimating the parameters of a robust loss formulation. Our methods have properties that are well-suited for dealing with image sequences containing several thousand flow vectors in real time. In particular, we demonstrate that the ERL method adds robustness without requiring costly iterative reweighting, resulting in very little runtime overhead.

Other methods for monocular odometry augment velocity data with planar homography estimates (Geiger et al. 2011; Song et al. 2013) or depth filters (Forster et al. 2014) to estimate scale. In this work, we do not rely on ground-plane estimation in order to maintain applicability to cases such as UAV navigation, where image sequences do not always contain the ground plane. Because we focus on frame-by-frame motion estimation, we cannot rely on a filtering approach to estimate depth. Our method can be augmented with domain-appropriate scale or depth estimators as part of a larger

Algorithm 1 ERL confidence weight estimation

Input: Measured flow $\{u_n\}_{n=1}^N$, sampled translational velocities $\{t_m\}_{m=1}^M$

Output: Estimated confidence weights $\{\hat{w}_n\}_{n=1}^N$

for all m **do**

 Compute scaled residuals:

$$\tilde{r}_u = |A^\perp(t_m)^\top (B\hat{\omega}_m(t_m) - u)|$$

 Compute maximum likelihood estimators of residual distribution:

$$\hat{\mu}_m = \text{median}(\tilde{r}_u)$$

$$\hat{b}_m = \frac{1}{N} \sum_{n=1}^N \|\tilde{r}_{u_n} - \hat{\mu}_m\|$$

end for

for all n **do**

 Compute confidence weights as expected likelihood under Laplacian fits:

$$\hat{w}_n = \frac{1}{M} \sum_{m=1}^M \mathcal{L}(\tilde{r}_{u_n}; \hat{\mu}_m, \hat{b}_m)$$

end for

return $\{\hat{w}_n\}_{n=1}^N$

SLAM system.

2.2.3. Robust optimization

In this work, we propose to increase the robustness of monocular egomotion estimation (i) by estimating each flow vector’s confidence weight as its expected residual likelihood (ERL) and (ii) by using a lifted robust kernel to jointly estimate confidence weights and model parameters. ERL confidence weights are conceptually similar to the weights recovered in the IRLS method for optimizing robust kernels (Holland and Welsch 1977). Robust kernel methods attempt to minimize the residuals of observations generated by the target model process (“inliers”) while limiting the influence of other observations (“outliers”). Such methods have been used very successfully in many domains of computer vision (Geman and Reynolds 1992; Black and Rangarajan 1996). However, we are unaware of any previous work that attempts to estimate confidence weights based on the distribution of residuals at counterfactual model parameters, as we do in the ERL method.

The lifted kernel approach offers another method to design and optimize robust kernels in particularly desirable ways. Lifted kernels have recently been used in methods for bundle adjustment in SfM (Zach 2014), object pose recovery (Zach et al. 2015), and non-rigid object reconstruction (Zollhöffer

et al. 2014). Our lifted kernel approximates the truncated quadratic loss, which has a long history of use in robust optimization in computer vision (Blake and Zisserman 1987) and has demonstrated applicability in a wide variety of problem domains.

Previous studies have used robust loss functions for monocular egomotion (Zhang and Tomasi 1999), visual SLAM (Newcombe et al. 2011), and RGB-D odometry (Kerl et al. 2013). To our knowledge, we present the first application of lifted kernels for robust monocular egomotion. Noise is typically handled in odometry by using sampling-based iterative methods such as RANSAC, which makes use of a small number of points to estimate inlier sets (typically five or eight points in monocular methods). The use of a robust kernel allows us to derive our final estimate from a larger number of points. This is desirable because the structure of the problem of continuous monocular odometry admits fewer correct solutions when constrained by a larger number of input points, which can better reflect the complex depth structure of real scenes. Our robust methods allow us to take advantage of a large number of flow estimates, which, while noisy, may each contribute weakly to the final estimate.

2.3. Problem formulation and approach

In this section, we present the continuous formulation of the problem of monocular visual egomotion. We describe and motivate our approach for solving the problem in the presence of noisy optical flow. We then describe two methods for estimating the confidence weights for each flow vector in a robust formulation of the problem, as well as the pipeline we use to estimate camera motion and scene depth.

2.3.1. *Visual egomotion computation and the motion field*

In the continuous formulation, visual egomotion methods attempt to estimate camera motion and scene parameters from observed local image velocities (optical flow). The velocity of an image point due to camera motion in a rigid scene under perspective projection is given by

$$u(x_i) = \rho(x_i)A(x_i)t + B(x_i)\omega. \quad (2.1)$$

where $u_i(x_i) = (u_i, v_i)^\top \in \mathbf{R}^2$ is the velocity (optical flow) at image position $x_i = (x_i, y_i)^\top \in \mathbf{R}^2$, $t = (t_x, t_y, t_z)^\top \in \mathbf{R}^3$ is the camera's instantaneous translational velocity, $\omega = (\omega_x, \omega_y, \omega_z)^\top \in \mathbf{R}^3$ is the camera's instantaneous rotational velocity, and $\rho(x_i) = \frac{1}{Z(x_i)} \in \mathbf{R}$ is the inverse of scene depth at x_i along the optical axis. We normalize the camera's focal length to 1, without loss of generality. In the case of calibrated image coordinates,

$$A(x_i) = \begin{bmatrix} 1 & 0 & -x_i \\ 0 & 1 & -y_i \end{bmatrix}, \quad (2.2)$$

$$B(x_i) = \begin{bmatrix} -x_i y_i & 1 + x_i^2 & -y_i \\ -1 - y_i^2 & x_i y_i & x_i \end{bmatrix}. \quad (2.3)$$

This formulation is appropriate for the small-baseline case where point correspondences between frames can be treated as 2D motion vectors.

The goal of monocular visual egomotion computation is thus to estimate the six motion parameters of t and ω and the N values for ρ from N point velocities u induced by camera motion. t and ρ are multiplicatively coupled in equation (2.1) above, so t can only be recovered up to a scale. We therefore restrict estimates of t to the unit hemisphere, $\|t\| = 1$.

The full expression for the set of N point velocities can be expressed compactly as

$$u = A(t)\rho + B\omega. \quad (2.4)$$

where the expressions for $A(x)$, $B(x)$, and $\rho(x)$ for all N points are

$$A(t) = \begin{bmatrix} A(x_1)t & 0 & \dots & 0 \\ 0 & A(x_2)t & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A(x_N)t \end{bmatrix} \in \mathbf{R}^{2N \times N} \quad (2.5)$$

$$B = \begin{bmatrix} B(x_1) \\ B(x_2) \\ \vdots \\ B(x_N) \end{bmatrix} \in \mathbf{R}^{2N \times 3} \quad (2.6)$$

and the velocity and depth for each of the points are concatenated to form $u = (u_1^\top, u_2^\top, \dots, u_N^\top)^\top \in \mathbf{R}^{2N \times 1}$ and $\rho = (\rho(x_1), \rho(x_2), \dots, \rho(x_N))^\top \in \mathbf{R}^{N \times 1}$. We estimate camera motion and scene depth by minimizing the objective

$$\begin{aligned} \min_{t, \rho, \omega} E(t, \rho, \omega) &= \min_{t, \rho, \omega} L(r(t, \rho, \omega)) \\ &= \min_{t, \rho, \omega} \|A(t)\rho + B\omega - u\|_2^2. \end{aligned} \quad (2.7)$$

Here, $L(x) : \mathbf{R}^N \rightarrow \mathbf{R}$ is a loss function and $r(t, \rho, \omega) : \mathbf{R}^{N+6} \rightarrow \mathbf{R}^N$ is a residual function for the flow field depending on the estimated model parameters. We first describe the case of an unweighted residual function under a quadratic loss, which is suitable for the case of Gaussian noise.

Following Zhang and Tomasi 1999, we note that no loss of generality occurs by first solving this objective for ρ in the least-squares sense. Minimizing over ρ gives

$$\begin{aligned} \min_{t, \omega} \min_{\rho} \|A(t)\rho + B\omega - u\|_2^2 \\ = \min_{t, \omega} \|A^\perp(t)^\top (B\omega - u)\|_2^2, \end{aligned} \quad (2.8)$$

where $A^\perp(t)$ is the orthogonal complement to $A(t)$. This expression no longer depends on ρ and depends on t only through $A^\perp(t)^\top$, which is fast to compute due to the sparsity of $A(t)$ (see section II of the supplement for more details).

In the absence of noise, we could proceed by directly minimizing equation (2.8) in t and ω . In particular, given a solution for t , we can directly solve for ω by least squares in $O(N)$ time. In the noiseless case, we estimate t by optimizing

$$\min_t \|A^\perp(t)^\top (B\hat{\omega}(t) - u)\|_2^2, \quad (2.9)$$

where $\hat{\omega}(t)$ is the least-squares estimate of ω for a given t (see section IV of the supplement for more details). This method of estimating t , ρ , and ω was shown to be consistent in Zhang and Tomasi 2002. That is, in the absence of outliers, this method leads to arbitrarily precise, unbiased estimates of the motion parameters as the sample size increases.

2.3.2. Robust formulation

However, the manipulations introduced in equations (2.8) and (2.9) rely on least-squares solutions and are not stable in the presence of outliers. Accordingly, instead of directly solving equation (2.9), we propose to solve a robust form. To do so, we introduce a confidence weight for each flow vector $w_i(u_i) \in [0, 1]$ to give

$$\begin{aligned} & \min_t L(r(t, \hat{\omega}(t)), w) \\ &= \min_t \|w \circ A^\perp(t)^\top (B\hat{\omega}(t) - u)\|_2^2, \end{aligned} \quad (2.10)$$

where $w = (w(u_1), w(u_2), \dots, w(u_N))^\top \in [0, 1]^N$ is the vector of all weights, $r \in \mathbf{R}^N$ is the vector of residuals for the flow field at some estimate of t , and \circ is the Hadamard product.

Each entry $w(u_i)$ of w attempts to weight the corresponding data point u_i proportionally to its residual at the optimal model parameters $(\hat{t}, \hat{\rho}, \hat{\omega})$, reflecting the degree to which the point is consistent with a single generating function for the motion in the scene, possibly with Gaussian noise. In other words, it reflects the degree to which u_i is an inlier for the optimal model of camera motion in a rigid scene. This is equivalent to replacing the choice of $L(x) = x^2$ as the loss in equation (2.9) with a function that grows more slowly.

We introduce a method to directly estimate the confidence weights as the expected residual likelihood (ERL) for each flow vector given the distribution of residuals for the flow field at a range of model parameters consistent with the solution in equation (2.9). We interpret each weight in terms of an estimate of the validity of the corresponding point under the model: that is, as an estimate of

the point’s residual at the optimal model parameters in a noise-free context. We compare ERL to a method that replaces $L(x) = x^2$ in equation (2.9) with a lifted truncated quadratic kernel (Zach 2014) and jointly optimizes the confidence weights and model parameters. We demonstrate that ERL outperforms the lifted kernel approach on the KITTI dataset, and both of these approaches outperform existing methods for monocular egomotion computation.

2.3.3. Confidence weight estimation by expected residual likelihood

Here, we describe the ERL method for estimating the confidence weights in (6), and we demonstrate that this method provides a good estimate of the appropriate confidence weights in the case of optical flow for visual egomotion.

At the optimal model parameters, (t^*, ρ^*, ω^*) , the residuals for inlier points (i.e. correct flow vectors due to rigid motion) are distributed according to a normal distribution, reflecting zero-mean Gaussian noise. However, in the presence of outliers, a zero-mean Laplacian distribution provides a better description of the residual distribution (see Figure 21 in the appendix). Accordingly, we can fit a Laplacian distribution to the observed residuals at the optimal model parameters to approximate the probability density function for residuals.

We use this property to identify outliers as those points that are inconsistent with the expected residual distribution at a range of model values. For each point, we compute the likelihood of each observed, scaled residual as

$$p(\tilde{r}_{u_i}^m | (t_m, \rho_m, \omega_m), \tilde{r}_u^m) = \mathcal{L}(\tilde{r}_{u_i}; \hat{\mu}_m, \hat{b}_m), \quad (2.11)$$

where $\tilde{r}_{u_i}^m$ is the scaled residual under the m^{th} model (t_m, ρ_m, ω_m) at the i^{th} flow vector and $\tilde{r}_u^m = (\tilde{r}_{u_1}^m, \tilde{r}_{u_2}^m, \dots, \tilde{r}_{u_N}^m)^\top$. We fit $\hat{\mu}_m$ and \hat{b}_m , respectively the location and scale parameters of the Laplacian distribution, to the set of scaled residuals \tilde{r}_u^m using maximum likelihood.

Because inliers exhibit smaller self-influence than outliers (Huber 2011), inlier residuals will typically be associated with higher likelihood values. However, the distribution used to estimate the likelihood reflects both the inlier and outlier points. If the counterfactual model parameters used to estimate

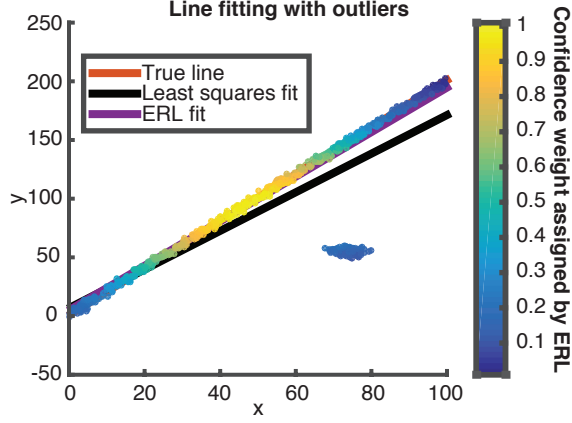


Figure 2: A 2D line-fitting problem demonstrating how ERL weights inliers and outliers. Inliers are generated as $y_i \approx 2x_i + 1$ with Gaussian noise. Each data points is colored according to its estimated confidence weight.

the m^{th} likelihood correspond to a model that is highly suboptimal, some outliers may be assigned higher likelihoods than they would be at the optimal model. Moreover, the presence of Gaussian noise means that the estimated likelihood for individual inliers may be erroneously low by chance for a particular model even if the optimal exponential distribution is exactly recovered.

To arrive at more reliable estimates and to discount the effect of erroneous likelihoods due to the specific model parameters being evaluated, we estimate the expected residual likelihood for each data point by evaluating the likelihood under M models,

$$\hat{w}_i = \mathbb{E}[\tilde{r}_{u_i}^m] = \frac{1}{M} \sum_{m=1}^M \mathcal{L}(\tilde{r}_{u_i}; \hat{\mu}_m, \hat{b}_m). \quad (2.12)$$

This method returns a vector $\hat{w} \in \mathbf{R}^N$. To use \hat{w} as confidence weights in a robust optimization context, we scale them to the interval $[0, 1]$. Scaling the maximum \hat{w}_i to 1 and the minimum \hat{w}_i to 0 for each flow field works well in practice.

The full process to estimate weights by ERL is shown in Algorithm 1. This method returns confidence weights in $O(MN)$ time, where M is set by the user. Empirically, the ERL method gives results that reflect the inlier structure of the data with small values of M (we use $M \approx 100$), allowing very

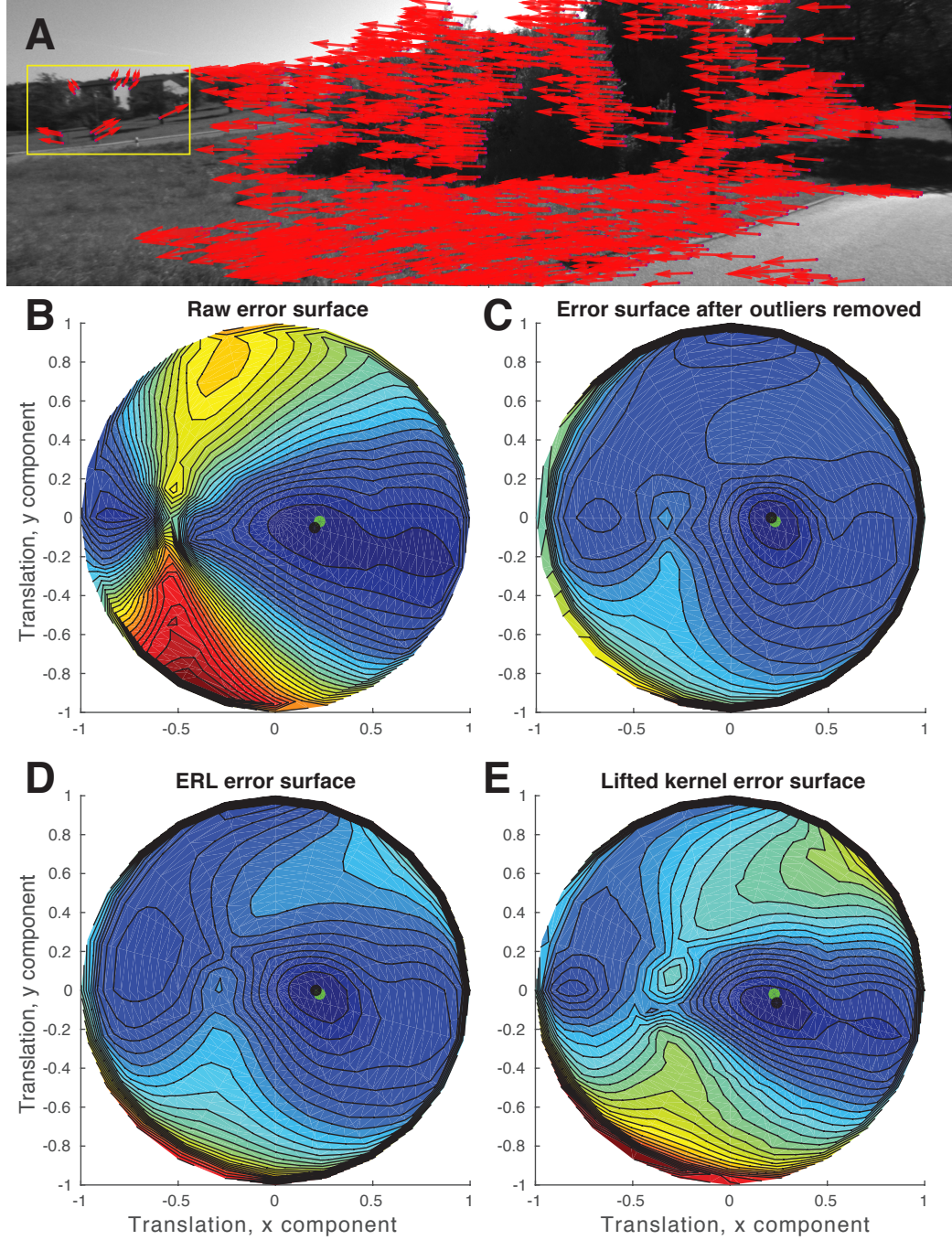


Figure 3: Robust methods recover the error surface of the outlier-free flow field. (A) Example optical flow field from two frames of KITTI odometry (sequence 10, images 14-15). Note the prominent outliers indicated by the yellow box. Error surfaces on this flow field for (A) the raw method (equation (2.9)) with all flow vectors, (B) with outliers removed by hand, and (C) with confidence weights estimated by ERL or (D) the lifted kernel. The green point is the true translational velocity and the black point the method's estimate. Blue: low error. Red: high error. Translation components are given in calibrated coordinates.

quick runtimes. In practice, the method assigns high weights to very few outliers while assigning low weights to acceptably few inliers. Thus, the method balances a low false positive rate against a moderately low false negative rate. This is a good strategy because our method takes a large number of flow vectors as input, which leads to redundancy in the local velocity information. Figure 2 illustrates the ERL method’s use in a simple 2D robust line-fitting application.

As discussed above, choosing values for the confidence weights in a least squares objective is equivalent to fitting a robust kernel. We note that regression under the assumption of Laplacian noise leads to an L1 cost. However, we have no guarantees about the form of the robust kernel corresponding to the weights chosen by the ERL method. Accordingly, we also explored using a robust kernel with known properties.

2.3.4. Robust estimation using a lifted kernel

Here, we explore the effect of jointly optimizing the confidence weights, $w(u)$, and ω for a given value of t using the lifted kernel approach described in Zach 2014. In our case, a lifted kernel takes the form

$$\begin{aligned} & \min_{t, \omega, w} \hat{L}(r(t, \omega), w) \\ &= \min_t \min_{\omega, w} (\|w \circ A^\perp(t)^\top (B\omega(t) - u)\|_2^2 + \sum_{i=1}^N \kappa^2(w_i^2)), \end{aligned} \quad (2.13)$$

where the lifted kernel of the loss L is denoted as \hat{L} . $\kappa(x) : \mathbf{R} \rightarrow \mathbf{R}$ is a regularization function applied to the weights. Because this approach does not rely on the least squares solution for rotational velocity, $\hat{\omega}$, it may gain additional robustness to noise. This approach also allows us to estimate the confidence weights for particular values of t , unlike the ERL approach, which relies on estimates at several values of t to produce stable results.

Different choices of κ produces different kernels. We use

$$\kappa(w^2) = \frac{\tau}{\sqrt{2}}(w^2 - 1), \quad (2.14)$$

which gives a kernel that is a smooth approximation to the truncated quadratic loss (Zach 2014). τ is

a hyperparameter that determines the extent of the quadratic region of the truncated quadratic loss. We set $\tau = 0.05$ for all results shown here, but other choices give similar results.

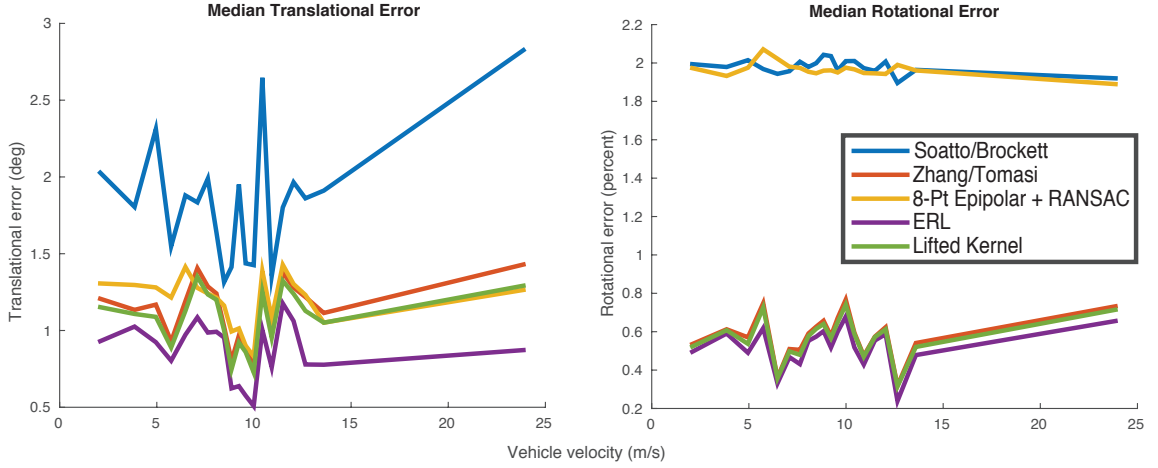


Figure 4: Median translational and rotational errors on the full KITTI odometry dataset for our methods and baselines.

The lifted kernel approach to solving nonlinear least squares problems is similar to IRLS insofar as it incorporates confidence weights on each of the data points and optimizes the values of these weights in addition to the value of the target model parameters. However, rather than alternately estimating the best weights given estimated model parameters and the best model parameters given estimated weights, the lifted approach simultaneously optimizes for both weights and model parameters, effectively “lifting” a minimization problem to a higher dimension.

The lifted kernel approach has several properties that are particularly beneficial for encouraging fast convergence. First, by using the weights to increase the dimensionality of the optimization problem, the lifted kernel minimizes the extent of regions of low gradient in the cost function. This ensures the method can quickly and reliably converge to minima of the function. Second, optimization can exploit the Gauss-Newton structure of the joint nonlinear least-squares formulation for faster convergence than the slower iterative-closest-points-like convergence exhibited by IRLS.

To illustrate the effect of our two robust optimization strategies, we display the error surfaces for the ERL and lifted-kernel methods on a sample flow field from KITTI (Figure 3). The error surfaces are

shown as a function of the translational velocity. Both methods recover error surfaces that resemble the error due to inlier flow vectors. The confidence weights estimated by ERL generally more closely resemble the pattern of inliers and outliers in flow data. To produce the results for the case with outliers removed, we strengthened the maximum bidirectional error criterion for flow inclusion to eliminate noisy matches and manually removed obvious outliers from the flow field.

2.4. Experiments

We compare the performance of the proposed methods (called “ERL” and “Lifted Kernel” in the figures) to several baseline methods for monocular egomotion/visual odometry from the literature: 5-point epipolar+RANSAC (using Stewenius et al. 2006)), 8-point epipolar+RANSAC (using Corke 2011)), and two continuous epipolar methods - Zhang/Tomasi (Zhang and Tomasi 1999), which is identical to equation (2.9), and Soatto/Brockett (Soatto and Brockett 1998). All experiments were run on a desktop with an Intel Core i7 processor and 16 GB of RAM. A single CPU core was used for all experiments.

With ~ 1000 flow vectors, the ERL method runs at 30-40 Hz in an unoptimized C++ implementation. Because of the low overhead of the ERL procedure, this is effectively the same runtime as the Zhang/Tomasi method. The lifted kernel optimization has no convergence guarantees, and it typically runs at < 1 Hz in a MATLAB implementation. Note that both of these runtimes can be significantly improved with better optimization. The Soatto/Brockett method runs extremely quickly (> 500 Hz), but performs poorly on real sequences. The implementation of epipolar+RANSAC used here runs at ~ 25 Hz. Optical flow for all our results was extracted using a multiscale implementation of the KLT method (Lucas and Kanade 1981; Tomasi and Kanade 1991).

For both ERL and the lifted approach, we optimize t using Gauss-Newton. We initialize t at a grid of values spaced over the unit hemisphere to decrease the chance of converging to a non-global minimum. We then prune the grid to a single initial value t_0 by choosing the grid point that gives the lowest residual under equation (2.10) or (2.13) for ERL or the lifted kernel, respectively. We then optimize to convergence starting from t_0 . This pruning strategy is effective at avoiding local minima

because good estimates for the weights return an error surface that is very similar to the noiseless case (see Figure 3) and this error surface is smooth with respect to the sampling density we use (625 points) (Chiuso et al. 2000). Confidence weights for ERL are computed using model parameters sampled on a coarser grid (100 points), as this is adequate to give good confidence weight estimates.

For all tests using the lifted kernel, we optimize the expression in equation (2.13) using the efficient Schur compliment implementation of Levenberg-Marquardt described in Zach 2014. Details of the optimization procedure used here are given in section III of the supplement. We did not explore jointly optimizing over t , ω , and w , but joint optimization over these model parameters with a lifted kernel is possible, and we plan to explore its use in future work.

2.4.1. Evaluation on KITTI

We evaluate the performance of our method using the KITTI dataset (Geiger et al. 2012), which is a collection of real-world driving sequences with ground-truth camera motion and depth data. The sequences contained in the dataset are challenging for state-of-the-art odometry methods for several reasons. First, they contain large inter-frame motions and repetitive scene structures that make estimating accurate flow correspondences difficult in real time. Second, several sequences feature little to no camera motion, which typically causes monocular odometry methods to fail. Finally, some sequences contain independent motion due to other vehicles and pedestrians, which violates the assumption of scene rigidity and makes reliable odometry more difficult.

All results are performed on neighboring frames of the KITTI odometry dataset (no skipped-frame sequences are evaluated), as these image pairs better match the modeling assumptions of continuous egomotion/odometry methods. All sequences were captured at 10 Hz at a resolution of 1392 x 512 pixels. We evaluated all methods on all 16 sequences of the KITTI odometry test set.

The results for methods on KITTI are shown in Figures 4, 5, and 6. For ease of visualization, the results for the 5-point epipolar method with RANSAC are not shown (they were significantly worse than all other methods we attempted). ERL produces the best estimates of translational velocity, while the lifted kernel produces results of similar quality to 8-point epipolar with RANSAC and

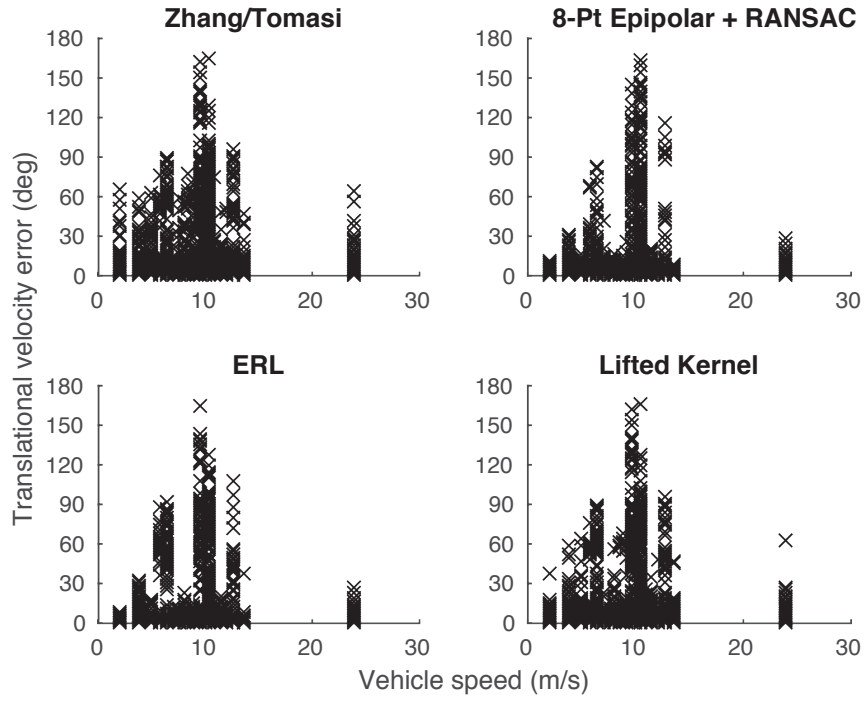


Figure 5: Full distribution of translational velocity errors.

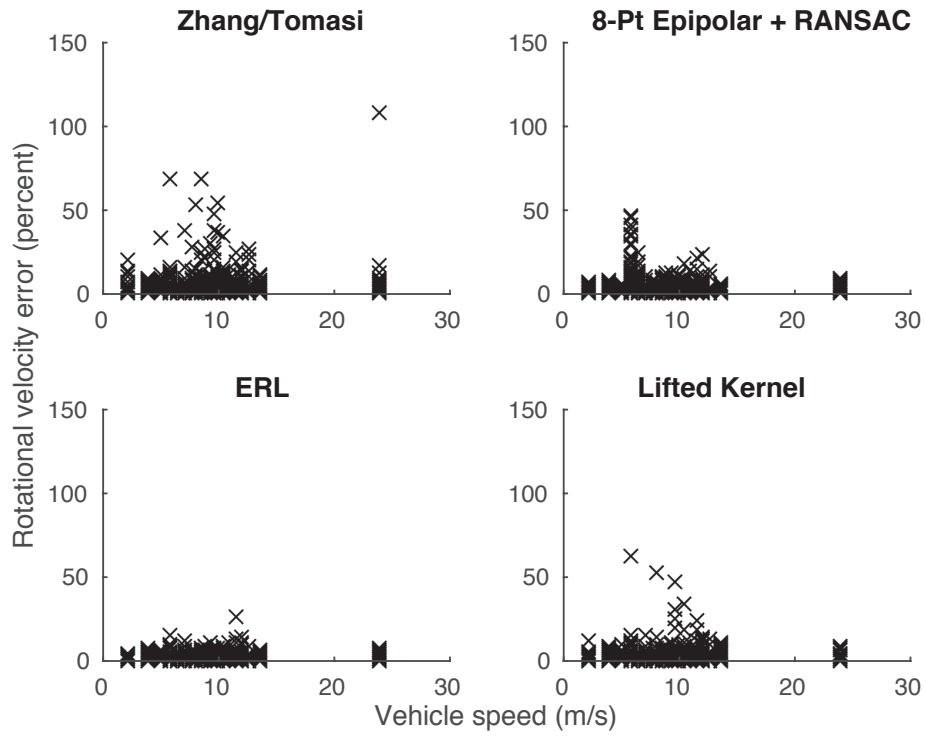


Figure 6: Full distribution of rotational velocity errors.

the Zhang/Tomasi method. ERL, the lifted kernel, and Zhang/Tomasi produce rotational velocity estimates of similar quality. The 8-point epipolar method produces worse estimates in this case because of the large baseline assumption, which is not suitable for rotational velocity estimation under these conditions. Soatto/Brockett produces bad estimates in these test cases because of the bias introduced by its algebraic manipulation.

2.4.2. *Synthetic sequences*

To estimate the robustness of our methods to outliers, we test the methods on synthetic data. Synthetic data were created by simulating a field of 1500 image points distributed uniformly at random depths between 2 and 10 m in front of the camera and uniformly in x and y throughout the frame. A simulated camera is moved through this field with a translational velocity drawn from a zero-mean Gaussian with standard deviation of 1 m/frame and a rotational velocity drawn from a zero-mean Gaussian with standard deviation of 0.2 radians/frame. Flow was generated from the resulting 3D point trajectories by perspective projection using a camera model with a 1 m focal length. All flow vectors were corrupted with noise in a random direction and magnitude drawn from a zero-mean Gaussian with a standard deviation $1/10^{\text{th}}$ the mean flow vector magnitude. Outliers were created by replacing a fraction of the points with random values drawn from a Gaussian fit to the magnitude and direction of all inlier flow vectors. We ran 100 iterations at each outlier rate. We ran all egomotion methods on the same data.

The errors in translational motion estimated on this data are shown in Figure 7. As expected, the two robust methods outperform least-squares methods for reasonable numbers of outliers. At higher outlier rates, however, the performance of both robust methods deteriorates. Interestingly, the performance of the lifted kernel method is stable even when the majority of data points are outliers. We are uncertain why the lifted kernel performs better than ERL on synthetic data, while the opposite is true for KITTI. This difference may be due to the way the data were generated - in KITTI, outliers often reflect real structures in the scene and may contain some information about camera motion, but this is not the case in the synthetic data. The difference may also be due in part to the difference in depth structures in KITTI and the synthetic data. In KITTI, flow magnitude for both inliers and

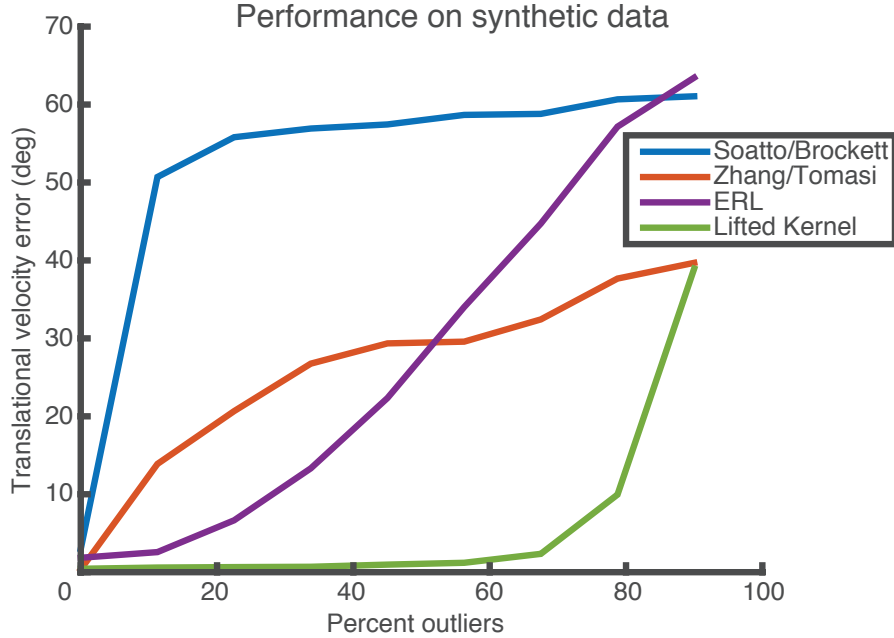


Figure 7: Translation error as a function of percent outliers on synthetic data for our robust methods and two baseline continuous egomotion methods.

outliers is reflective of depth structure, and depth in real scenes is not distributed uniformly.

2.5. Conclusions

We have introduced new techniques for robust, continuous egomotion computation from monocular image sequences. We described ERL, a novel robust method that directly estimates confidence weights for the vectors of a flow field by evaluating the distribution of flow residuals under a set of self-consistent counterfactual model parameters. We also introduced a new formulation of the perspective motion equation using a lifted kernel for joint optimization of model parameters and confidence weights. We compared the results of ERL and the lifted kernel formulation, and showed that while the lifted kernel appears to be more stable in the presence of a large fraction of outliers, ERL performs better in a real-world setting. The ERL method achieves good results on KITTI without relying on stereo data or ground plane estimation and accordingly is well-suited for use in lightweight UAV navigation. We are unable to directly evaluate our methods on this target domain because there are currently no UAV datasets with suitable ground truth. Although the empirical

results here are promising, we have no guarantees on the weights recovered by ERL, and this remains a topic for future work.

Our code is publicly available at https://github.com/stephenphillips42/erl_egomotion.

CHAPTER 3 : Understanding image motion with group representations

3.1. Introduction

Motion perception is a key component of biological and computer vision. By understanding how a stream of images reflects the motion of the world around it, an agent can better judge how to act. For example, a fly can use visual motion cues to dodge an approaching hand and to distinguish this threat from a looming landing surface (Reiser and Dickinson 2013). Motion is an important cue for understanding actions and predicting 3D scene structure, and it has been extensively studied from computational, ethological, and biological perspectives (Hildreth and Koch 1987).

In computer vision, the problem of motion representation has typically been approached from either a local or global perspective. Local representations of motion are exemplified by optical flow. Flow represents image motion as the 2D displacement of individual pixels of an image, giving rich low-level detail while foregoing a compact representation of the underlying scene motion. In contrast, global representations such as those used in visual odometry attempt to compactly explain the movement of the whole scene. Such representations typically rely on a rigid world assumption, thus limiting their applicability to more general settings.

Image transformations due to motion form a subspace of all continuous image transformations. Smooth changes in the motion subspace correspond to sequences of images with realistic motion. We wish to characterize this subspace. The motion subspace differs from other image transformation subspaces, such as changes in the space of images of human faces. Smooth changes in this space also form a subspace of image transformations, but one containing transformations that do not occur in natural image sequences, such as the face of one person transforming into the face of another. A representation that characterizes motion should be sensitive to the distinction between image transformations that are realistic (produced by image motion) vs. those that are unrealistic (not produced by image motion).

To be useful for understanding and acting on scene motion, a representation should capture the

motion of the observer and all relevant scene content. Supervised training of such a representation is challenging: explicit motion labels are difficult to obtain, especially for nonrigid scenes where it can be unclear how the structure and motion of the scene should be decomposed. We propose a framework for learning global, nonrigid motion representations without labels. While most methods of representing motion rely on pixel-level reconstruction or correspondence to guide learning, our method constrains the representation itself by directly addressing the properties of the latent motion space.

Motion has several properties that we use to operationalize to what extent a model characterizes it. (1) A model of motion can be read out to estimate metric properties of the motion in the scene, such as the camera translation and rotation. (2) A model of motion should represent the same motion identically regardless of the image content. For example, the motion of an object moving to the right should be represented the same whether the object is a cat or a dog. (3) A model of motion should distinguish sequences produced by natural motion from sequences with image transitions not produced by natural motion, such as cuts.

Here, we present a general model of visual motion and describe how the group properties of visual motion can be used to constrain learning in this model (Figure 8). We enforce the group properties of associativity and invertibility during training using a metric learning approach (Chopra et al. 2005) on recomposed sequences. We describe how this technique can be used to train a deep neural network to represent the motion in image sequences of arbitrary length in a low-dimensional, global fashion. We present evidence that the learned representation captures the global structure of motion in both 2D and 3D settings without labels, hard-coded assumptions about the scene, or explicit feature matching.

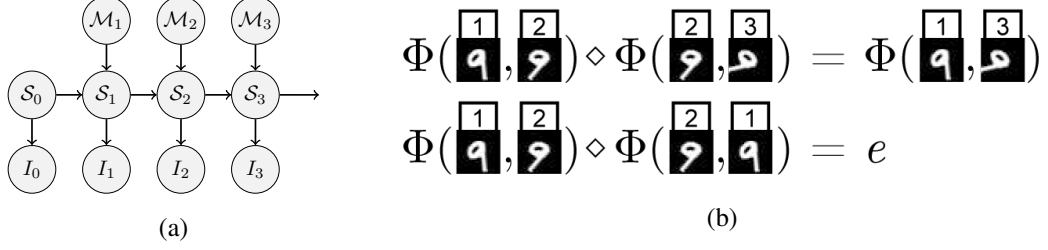


Figure 8: **(a)** A graphical model describing the relationship between the latent scene structure $\{\mathcal{S}_t\}$, motion $\{\mathcal{M}_t\}$, and the observed images of a sequence. We describe a method for learning a representation $\overline{\mathcal{M}}$ of the motion space \mathcal{M} from observed image sequences $\{I_t\}$. **(b)** By recomposing sequences of images to satisfy the group properties of associativity and invertibility, we construct pairs of image sequences with equivalent motion. We use these properties to learn an approximate group homomorphism $\Phi \in \overline{\mathcal{M}}$ between motion in the world and in an embedding.

3.2. Related work

3.2.1. Motion representations

The most common global representations of motion are structure from motion (SfM) and simultaneous localization and mapping (SLAM), which represent motion as a sequence of poses in $\mathcal{SE}(3)$ perhaps along with a static point cloud (Scaramuzza and Fraundorfer 2011; Fraundorfer and Scaramuzza 2012). These approaches have achieved great success in many applications in recent years, but they are unable to represent non-rigid or independent motions. The most commonly used local representation is optical flow, which estimates pixel-wise motion over the image, typically constraining it with a smoothness prior (Sun et al. 2010). Scene flow (Wedel et al. 2008) and non-rigid structure from motion (Xiao et al. 2004) represent a larger class of 3D motions by generalizing optical flow to the estimation of 3D point trajectories. These methods represent motion only at local regions (typically single points) and do not attempt to compactly capture the overall motion.

More similar to our approach is work designing or learning spatio-temporal features (STFs) (Laptev 2005). STFs are localized and flexible enough to represent non-rigid motions. They typically include a dimensionality reduction step and hence are somewhat global in purview. Recent work has used convolutional neural nets (CNNs) to learn task-related STFs directly from images, including Tran et al. 2015 and Le 2013. Unlike our work, both of these approaches are restricted to fixed temporal

windows of representation. Taylor et al. 2010 uses a standard unsupervised learning technique to learn spatiotemporal features useful for action recognition but not for motion itself.

3.2.2. Learning representations using visual structure

Several recent works have used knowledge of the geometric or spatial structure of images or scenes to train representations. Doersch et al. 2015 trains a CNN to classify the correct configuration of image patches to learn the relationship between an image’s patches and its semantic content. The resulting representation can be fine-tuned for image classification. Yu et al. 2016 and Pătrăucean et al. 2016 train networks to estimate optical flow using the brightness constancy assumption and a smoothness prior as a learning signal. Zhu et al. 2017 and Ren et al. 2017 learn flow using a similar technique. As with other flow based methods, these works use photometric, local constraints. Garg et al. 2016 uses the relationship between depth and disparity to learn to estimate depth from a rectified stereo camera pair with a known baseline. Similarly, Konda and Memisevic 2014 treats motion as a latent variable and exploits the relationship between motion and depth to estimate depth.

Other works that learn from sequences typically focus on static image content rather than motion. Of these, the most similar to ours is Misra et al. 2016, which shuffles the order of images in a sequence to learn representations of image content. Their approach is designed to capture single image properties that are correlated with temporal order rather than motion itself and their shuffling procedure does not preserve the group properties forming the basis of our learning technique. A related approach is slow feature analysis (Wiskott and Sejnowski 2002), which is motivated by the notion that slowly varying latents are often behaviorally relevant. Other works exploring learning from sequences include Jayaraman and Grauman 2015, which learns a representation equivariant to the egomotion of the camera, and Agrawal et al. 2015, which learns to represent images by explicitly regressing the egomotion between two frames. Instead of learning to represent motion, these works use labeled motion as a learning cue.

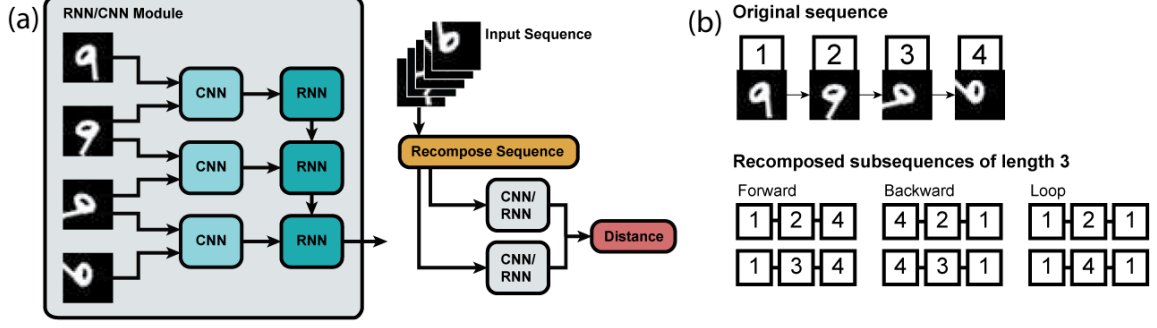


Figure 9: **(a)** Network structure. The RNN output at the final step of the sequence is treated as the sequence embedding. During training, the distance between sequence embeddings is adjusted using an embedding loss. **(b)** We recompose sequences to enforce associativity and invertibility. Sequences with equivalent motion (e.g. 1-2-4 and 1-3-4) serve as positive examples, while sequences with inequivalent motions (e.g. 1-2-4 and 4-3-1) serve as negative examples.

3.3. Approach

3.3.1. Group properties of motion

We base our method on the observation that the set of 3D scene motions, equipped with the composition operation, forms a group. This group describes the latent structure of transformations in continuous, real-world image sequences. By learning an embedding that captures the transformations in scenes that occur during motion, we approximate a group homomorphism between the latent motion of the scene and a representation of this motion. We design our method to capture associativity and invertibility, which allows us to reason about how motions relate and can be composed.

To see that a latent motion space respects these properties, first consider a latent structure space \mathcal{S} . In this model, an element of the structure space generates images \mathcal{I} via a projection operator $\pi : \mathcal{S} \rightarrow \mathcal{I}$. We also define a latent motion space \mathcal{M} , which is some closed subgroup of the set of homeomorphisms on \mathcal{S} . For any element S of the structure space \mathcal{S} , a continuous motion sequence $\{M_t \in \mathcal{M} \mid t \geq 0\}$ generates a continuous image sequence $\{I_t \in \mathcal{I} \mid t \geq 0\}$ where $I_t = \pi(M_t(S))$. For a discrete set of images, we can rewrite this as $I_t = \pi((M_{\Delta_t} \circ M_{t-1})(S)) = \pi(M_{\Delta_t}(S_{t-1}))$, which defines a hidden Markov model, as illustrated in Figure 8 (a). As \mathcal{M} is a closed subgroup of the homeomorphisms on \mathcal{S} , it is associative, it contains the identity, and all of its elements have unique

inverses in the group.

A simple, specific case of this model is rigid image motion, such as the motion produced by a camera translating and rotating through a rigid scene in 3D. Here, the latent structure of the scene \mathcal{S} can be modeled by a point cloud with a motion space given by $\mathcal{M} = \mathcal{SE}(3)$. For a scene with N rigid bodies, we can describe the motion with a tuple of $\mathcal{SE}(3)$ values, $\mathcal{M} = (\mathcal{SE}(3))^N$, where the N th motion acts on the set of points belonging to the N th rigid body. Generalizing \mathcal{M} to general homeomorphisms gives the most general case of motion. As different scenes contain different degrees of freedom and affordances, it is typically unclear which group effectively characterizes motion in a given real-world setting. We propose to learn this in a data-driven manner.

3.3.2. Learning motion by group properties

Our goal is to learn a function $\Phi : \mathcal{I} \times \mathcal{I} \rightarrow \overline{\mathcal{M}}$ that maps pairs of images to a representation $\overline{\mathcal{M}}$ of the motion space \mathcal{M} . We also learn a corresponding composition operator $\diamond : \overline{\mathcal{M}} \rightarrow \overline{\mathcal{M}}$ that emulates the composition of elements in \mathcal{M} . This representation and operator should respect the properties of the motion group in question.

We exploit the structure of the domain to learn to represent and compose motions without labels. If an image sequence $\{I_t\}$ is sampled from a continuous motion sequence, then the sequence representation should have the following properties for all times t_0, t_1, t_2, t_3 , where $t_0 < t_1 < t_2 < t_3$, reflecting the group properties of the latent motion:

- (i) Associativity: $\Phi(I_{t_0}, I_{t_2}) \diamond \Phi(I_{t_2}, I_{t_3}) = (\Phi(I_{t_0}, I_{t_1}) \diamond \Phi(I_{t_1}, I_{t_2})) \diamond \Phi(I_{t_2}, I_{t_3}) = \Phi(I_{t_0}, I_{t_1}) \diamond (\Phi(I_{t_1}, I_{t_2}) \diamond \Phi(I_{t_2}, I_{t_3})) = \Phi(I_{t_0}, I_{t_1}) \diamond \Phi(I_{t_1}, I_{t_3})$. The motion of differently composed subsequences of a sequence are equivalent.
- (ii) Existence of the identity element: $\Phi(I_{t_0}, I_{t_1}) \diamond e = \Phi(I_{t_0}, I_{t_1}) = e \diamond \Phi(I_{t_0}, I_{t_1})$, and $e = \Phi(I_t, I_t)$ for any t . Null image motion corresponds to the (unique) identity in the latent space.
- (iii) Invertibility: $\Phi(I_{t_0}, I_{t_1}) \diamond \Phi(I_{t_1}, I_{t_0}) = e$. The motion of a reversed image sequence is the inverse of the motion of the original image sequence.

We use an embedding loss to approximately enforce associativity and invertibility among subsequences sampled from an image sequence. Associativity is encouraged by pushing differently composed sequences with equivalent motion to the same representation. Invertibility of the representation is encouraged by pushing each forward sequence away from its backward counterpart and by pushing all loops to the same representation (i.e. to a learned representation of the identity in the embedding space). We encourage the uniqueness of the identity representation by pushing loops away from non-identity sequences in the representation. Because loops have equivalent (identity) motion regardless of scene content, we also push together loops drawn from different sequences. This procedure is illustrated schematically in Figure 9.

Learning in this framework can be viewed as inference on the graphical model in Figure 8 (a). Learning a representation of motion is an underconstrained problem, and the group learning rules we introduce here constrain the problem with minimal restriction on the types of scene changes that can be embedded.

In contrast, in optical flow, inference is constrained using the brightness constancy assumption, which assumes that the illumination of a projected scene point does not change between frames (Horn and Schunck 1981). Our framework encompasses flow inference if brightness constancy is viewed as a constraint on the projection operator π . However, the brightness constancy assumption is invalid in many settings. Our model’s assumptions about geometric properties of motion in the world are valid even over large motions and changing illumination.

The latent structure and motion of a scene are in general non-identifiable, which implies that for any given scene, there are several $\overline{\mathcal{M}}$ that can adequately represent \mathcal{M} . We do not claim to learn a unique representation of motion, but rather we attempt to capture one such representation. Our method assumes the scene has a relatively stable structure, and we do not expect it to handle rapidly changing content or sequence cuts. We also expect our method to have difficulty representing motion in cases of temporally extended occlusion due to the unobservability of motion in such settings.

3.3.3. Sequence learning with neural networks

The functions Φ and \diamond are implemented as a convolutional neural network (CNN) and a recurrent neural network (RNN), respectively. We use a long short-term memory (LSTM) RNN (Hochreiter and Schmidhuber 1997) due to its ability to reliably learn over long time sequences. The input to the network is in an image sequence $[I_1, \dots, I_t]$. The CNN Φ processes these images and outputs an intermediate representation $[C_{1,2}, \dots, C_{t-1,t}]$. The LSTM operates over the sequence of CNN outputs to produce an embedding sequence $[R_{1,2}, \dots, R_{t-1,t}]$. We treat $R(\{I_t\}) = R_{t-1,t}$ as the embedding of sequence $\{I_t\}$. This configuration is illustrated schematically in Figure 9 (a).

Table 1: Average embedding error (equation 3.1) on held-out data. Results are averaged over forward, backward, and loop sequences. Errors are relative to a chance error of 1: values lower than 1 indicate that equivalent (inequivalent) motions are close together (far apart) in the embedding space.

CNN input method	Motion condition	MNIST	KITTI
Image pairs	Equivalent	$8.1\text{e-}4$	$7.2\text{e-}3$
Image pairs	Inequivalent	$1.7\text{e-}2$	$8.0\text{e-}2$
Single image	Equivalent	0.74	$3.5\text{e-}2$
Single image	Inequivalent	0.79	3.5

The network is trained to minimize a hinge loss with respect to the embedding of pairs of sequences:

$$L(R^1, R^2) = \begin{cases} d(R^1, R^2), & \text{if positive pair} \\ \max(0, m - d(R^1, R^2)), & \text{if negative pair} \end{cases} \quad (3.1)$$

where $d(R^1, R^2)$ measures the distance between the embeddings of two example sequences $\{I_t^1\}$ and $\{I_t^2\}$, and m is a fixed scalar margin. Positive examples are image sequences that are compositionally equivalent, while negative examples are those that are not. We use the cosine distance for all experiments (with $m = 0.5$), as it is smooth and discourages learning the trivial embedding. In early experiments, results with an L2 distance were similar.

We include six recomposed subsequences for each training sequence: two forward, two backward, and two identity subsequences, as shown in Figure 9 (b). Subsequences are sampled such that all three sequence types share some of their frames. To discourage the network from paying attention to only the beginning or end of a sequence, we use several image recomposing schemes. Forward

and backward sequences are sampled to either have the same or different starting frames, and they are drawn from either the same subsequence or from temporally adjacent subsequences. Because the network is exposed to sequences with the same start and end frames but different motion, this sampling procedure encourages the network to rely on features in the motion domain, rather than on static differences. During training, we use sequences of varying length to encourage generalization to motions of different temporal scale.

We also explored learning a representation Φ taking single images (and not image pairs) as CNN input. Because the CNN in this configuration only has access to single images, it cannot extract image motion directly. In all domains we tested, the representation learned from image pairs outperformed the one learned from single images (Table 1).

3.4. Experiments

We first demonstrate that our learning procedure can discover the structure of motion in the context of rigid scenes undergoing 2D translations and rotations. We then show that our method learns features useful for representing motion on KITTI (Geiger et al. 2012), a dataset of vehicle sequences with motion due to the camera and independent objects. In all experiments, networks were trained using Adam (Kingma and Ba 2014). For MNIST training, we used a fixed decay schedule of 30 epochs and with a starting learning rate chosen by random search ($1e-2$ was a typical value). For MNIST, typical batch sizes were 50-60 sequences, and for KITTI (Geiger et al. 2012) the batch sizes were typically 25-30 sequences. All networks were implemented in Torch (Collobert et al. 2011).

We use dilated convolutions to obtain large receptive fields suitable for capturing large-scale motion patterns. We used ReLU nonlinearities and batch normalization (Ioffe and Szegedy 2015) after each convolutional layer. CNN output was passed to an LSTM with 256 hidden units, followed by a linear layer with 256 hidden units. In all experiments, CNN-LSTMs were trained on sequences 3-5 images in length. We tested MNIST networks with sequences of up to 12 images with similar results.

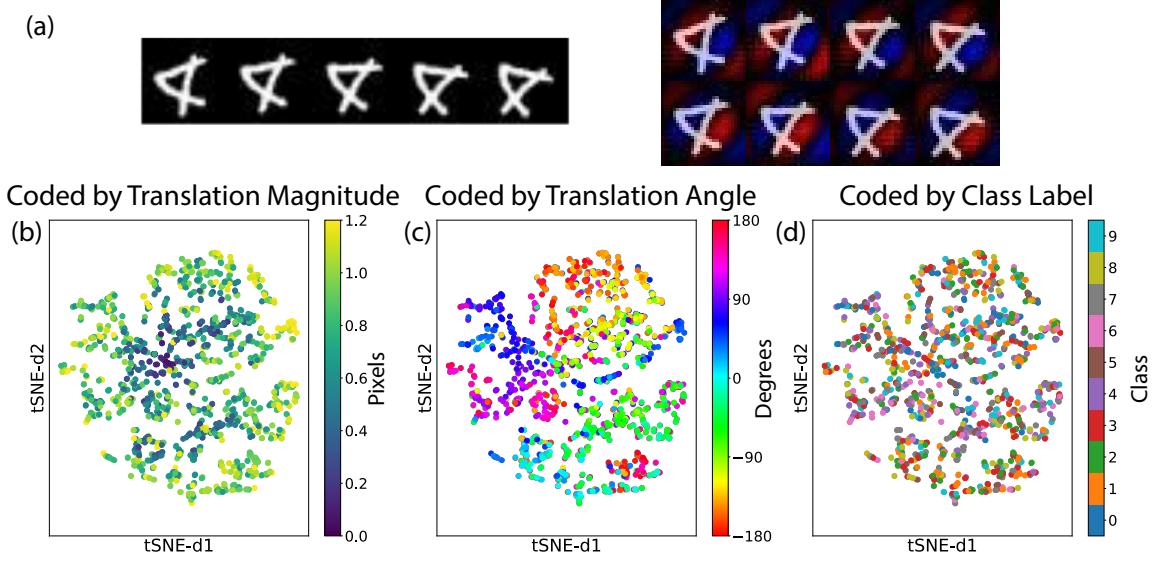


Figure 10: **(a)** An example test sequence from MNIST and the corresponding saliency maps. Saliencies show the gradient backpropagated from the final RNN timestep. Each column represents an image pair passed to one of the CNNs. **(b)-(d)** tSNE of the network embedding on the test set, with points labeled by **(b)** the magnitude of translation in pixels, **(c)** the translation direction in degrees, and **(d)** the digit label (0-9). The representation clusters sequences by both translation magnitude and direction but not identity.

3.4.1. Rigid motion in 2D

To test the ability of our learning procedure to represent motion, we trained a network on a dataset consisting of image sequences created from the MNIST dataset. Each sequence consists of images undergoing a smooth motion drawn from the group of 2D translations and rotations ($\mathcal{SE}(2)$) for 20 frames. We sampled transformation parameters uniformly from $[-10, 10]$ pixels for both horizontal and vertical translation and from $[0, 360)$ degrees for rotation. Validation errors are given in Table 1.

We visualize the representation learned on this data using tSNE (van der Maaten and Hinton 2008) on the sequence embedding for test images undergoing a random translation (Figure 10). The network representation clearly clusters sequences by both the direction and magnitude of translation. No obvious clusters appear in terms of the image content. Similar results were obtained when test data included both translation and rotation. This suggests that the network has learned a representation that captures the properties of motion in the dataset. This content-invariant clustering occurs even

Table 2: Linear regression from the learned embedding to the translation and rotation of the KITTI odometry dataset consistently performs better than chance (guessing the mean value). Table entries show mean squared error \pm standard error (percent improvement).

	Translation X (cm)	Translation Y (cm)	Translation Z (cm)
Mean	$5.92 \pm 1.5e-01$	$3.01 \pm 1.2e-01$	$1904.75 \pm 3.1e+01$
Ours	$5.05 \pm 1.2e-01$ (14.71%)	$2.83 \pm 1.2e-01$ (6.10%)	$1539.04 \pm 2.3e+01$ (19.20%)
Flow+PCA (4 PCs)	$3.18 \pm 9.5e-02$ (46.27%)	$2.92 \pm 1.2e-01$ (3.47%)	$1754.36 \pm 2.8e+01$ (7.91%)
Flow+PCA (256 PCs)	$1.89 \pm 8.5e-02$ (68.07%)	$2.32 \pm 1.2e-01$ (23.42%)	$239.46 \pm 5.6e+00$ (87.43%)
	Rotation X (deg)	Rotation Y (deg)	Rotation Z (deg)
Mean	$0.02 \pm 3.3e-04$	$0.98 \pm 1.6e-02$	$0.03 \pm 4.8e-04$
Ours	$0.02 \pm 3.1e-04$ (4.03%)	$0.79 \pm 1.5e-02$ (19.12%)	$0.02 \pm 4.0e-04$ (21.28%)
Flow+PCA (4 PCs)	$0.02 \pm 3.3e-04$ (1.16%)	$0.29 \pm 3.3e-03$ (70.11%)	$0.03 \pm 4.8e-04$ (0.17%)
Flow+PCA (256 PCs)	$0.00 \pm 9.8e-05$ (79.33%)	$0.05 \pm 1.7e-03$ (94.50%)	$0.01 \pm 1.3e-04$ (82.53%)

though the network was never trained to compare images with different spatial content and the same motion.

To further probe the network, we visualize image-conditioned saliency maps in Figure 10. These saliency maps show the positive (red) and negative (blue) gradients of the network activation with respect to the input image. As discussed in Simonyan et al. 2013, such a saliency map can be interpreted as a first-order Taylor expansion of the function Φ , evaluated at image I . The saliency map thus gives an indication of how pixel values affect the representation. These saliency maps show gradients with respect to the output of the LSTM over the sequence.

Intriguingly, these saliency maps bear a strong resemblance to the spatiotemporal energy filters of classical motion processing (Adelson and Bergen 1985), which are known to be optimal for 2D speed estimation in natural scenes under certain assumptions (Burge and Geisler 2015). We note that these saliency maps do not simply depict the shape of the filters of the first layers, but rather represent the implicit filter instantiated by the full network on this image. When compared across different frames, it becomes clear that the functional mapping learned by the network can flexibly adapt in orientation and arrangement to the image content, unlike standard energy model filters.

3.4.2. Real-world motion in 3D

We use the KITTI dataset (Geiger et al. 2012) to test the model’s representation of motion in 3D scenes with camera and independent motion. We use the representation trained on KITTI tracking

Table 3: Interpolation distances on KITTI (as in Figure 11), averaged across test data. Distances are consistently lower for the true frame than for visually similar frames (inside sequence) and dissimilar frames (outside sequence) when using the embedding, but not the Euclidean distance.

Method	Skipped frames	True middle frame	Inside (min value)	Outside (min value)
Embedding	1	3.91e-03	7.67e-02	2.94e-01
Euclidean	1	7.92e-04	7.97e-04	1.09e-03
Embedding	2	1.18e-02	2.02e-02	1.34e-01
Euclidean	2	9.59e-04	8.13e-04	1.08e-03

in all experiments. First, we evaluate how well it can decode camera motion. We compute the representation on all two-frame sequences of KITTI visual odometry, which are labeled with ground truth camera poses. We then linearly regress from these representations to the change in camera pose between the frames using least squares.

For comparison, we show results using a recent self-supervised flow algorithm (Yu et al. 2016). The output of this method is a dense optical flow field. In order to regress from this flow field to camera poses, we downsample the flow fields and run principal component analysis (PCA) over the full training set. We then linearly regress from the flow field PCA components to the camera motion parameters using least squares. Flow fields are computed at a resolution of 320x96 pixels, and PCA is computed on downsampled flow fields of size 160x48 pixels. We include up to 256 PCA components in the regression. We refer to this method as Flow+PCA.

Results on held-out test data are displayed in Table 2. Despite not being trained on any ground truth pose and not seeing any data from the odometry dataset, the learned representation decodes pose consistently better than chance (guessing the mean value). The largest improvements are in X and Z translation, which also exhibit the most variance in the KITTI odometry dataset. These results are not competitive with the Flow+PCA results or state-of-the-art odometry methods, but they suggest our method recovers a meaningful representation of motion. On KITTI visual odometry, our method performs similarly to regression to the Flow+PCA method using four to five principal components 22, which suggests that it is able to capture the dominant global components of motion.

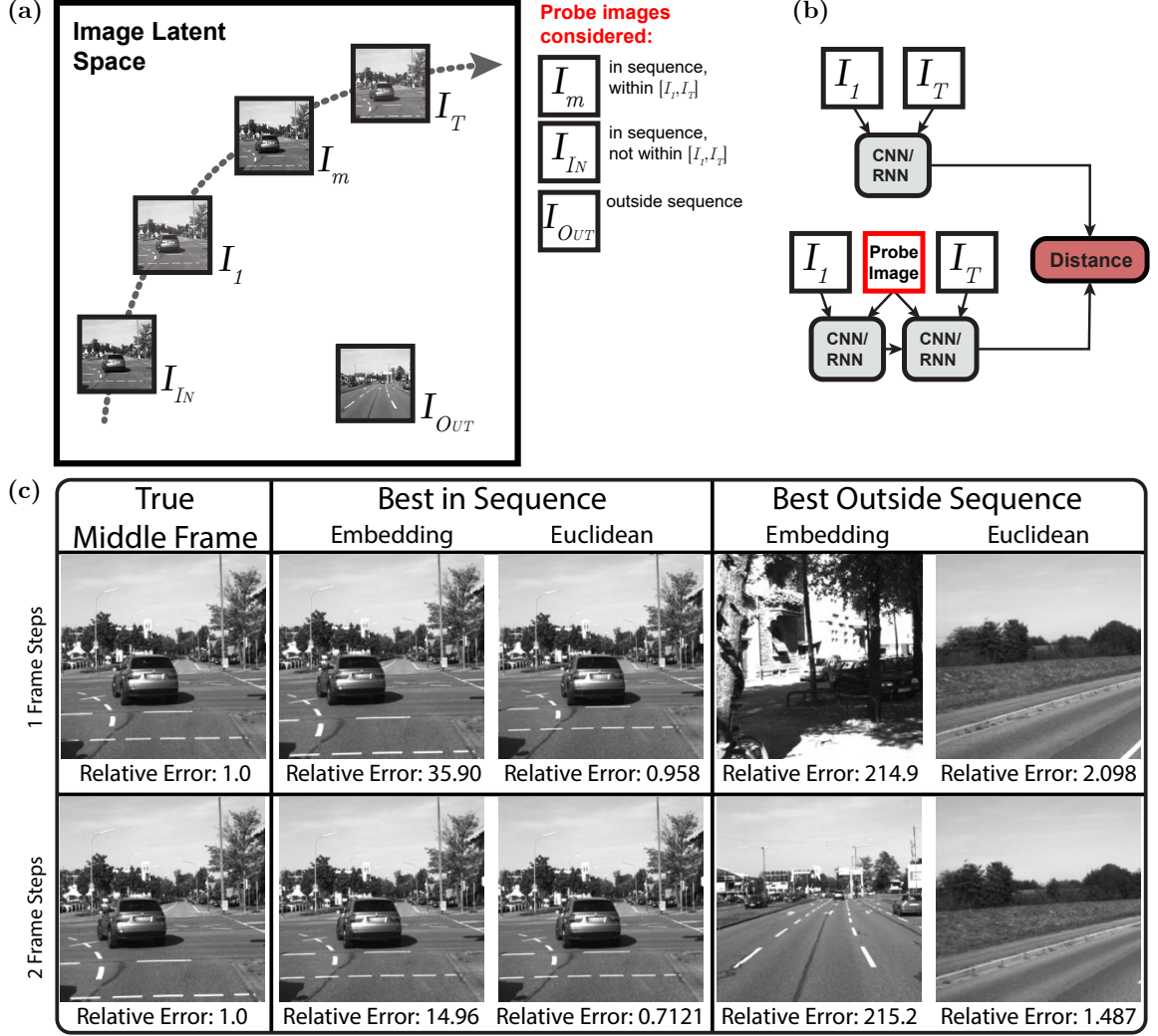


Figure 11: **(a)** Natural image motion is a subspace of the space of all image transformations, and a particular motion can be viewed as a path in the latent space of natural images. Although $[I_1, I_K, I_T]$ has a total transformation equivalent to $[I_1, I_T]$ for any value of K , only $[I_1, I_m, I_T]$ can be produced by natural image motion. **(b)** In interpolation experiments, we compare the distance between the embedding of $[I_1, I_T]$ with the embedding of this sequence after inserting either a true middle frame (I_m) or another frame (I_{IN} or I_{OUT}). **(c)** Images with lowest relative error taken from the sequence or from the whole dataset, for each distance measure. Errors are relative to that of the true middle frame in the corresponding measure: high relative errors ($\gg 1$) indicate the distance distinguishes realistic motion from unrealistic motion. Images other than true middle frame produce dramatically higher errors when using the embedding but not when using a Euclidean distance.

Next, we test the ability of our network to capture the typical motion of the scene by quantifying its performance on an interpolation task. Given an image sequence $[I_1, \dots, I_T]$, we compute the distance between the embedding of the first and last frames, $R([I_1, I_T])$, vs. the sequence composed of the first

frame, a middle frame, and the last frame $R([I_1, I_m, I_T])$ (Figure 11). By comparing the distance when using the true middle frame with the distance when using a different middle frame, we can estimate how sensitive the network is to deviations from the typical dynamics of natural scenes, and hence how well it has learned the relevant motion subgroup. Results for the full KITTI tracking dataset are shown in Table 3. We compare our method to a Euclidean distance, computed as the mean pixelwise distance between the probe image and either I_1 or I_T (whichever is smaller). Note that the embedding distance of the true frame is dramatically lower than that of all other frames. This does not hold for the Euclidean distance, which is often lower for non-interpolating images in the sequence, and is not dramatically different for frames from other sequences.

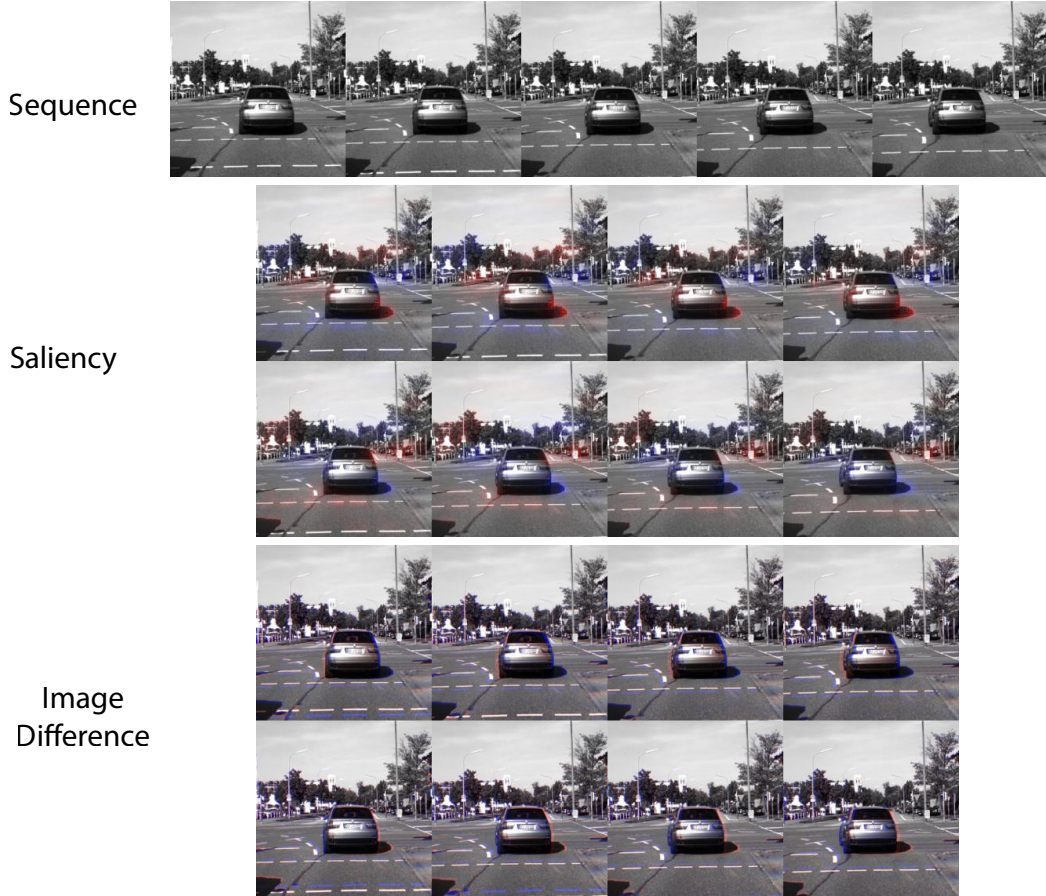


Figure 12: Saliency results on a test sequence from KITTI tracking with both camera and independent motion. The network focuses on areas that are relevant to determining motion in 3D, not simply regions with large temporal image gradients.

Finally, we visualize saliency maps on an example sequence in the KITTI dataset in Figure 12. The saliency map highlights objects moving in the background and the independent motions of the car in the foreground. The network highlights areas of the car that can move, such as the bumper, even when these areas don't contain prominent image differences. These results suggest our method may be useful for learning features for independent motion detection and tracking.

There are few standard tasks for directly evaluating motion methods beyond odometry. We attempted to regress from our learned representation to action classes but were unable to obtain competitive performance. This is not surprising: previous work has shown spatial features are more discriminative for this task, and motion features require extensive processing to be useful (e.g. Simonyan and Zissermann 2014). In future work, we will explore using group properties to encourage intermediate latents to represent motion along with other tasks. We expect that an embedding that maintains representations of spatial content alongside representations of motion will be more successful in settings like action recognition that depend on both sets of features.

3.5. Conclusion

We have presented a new model of motion and a method for learning motion representations. We have shown that enforcing group properties of motion is sufficient to learn a representation of image motion. These results suggest that this representation is able to generalize between scenes with disparate content and motion to learn a motion state representation useful for navigation, prediction, and other behavioral tasks relying on motion. Because of the wide availability of unlabeled video sequences in many settings, we expect our framework to be useful for generating better global motion representations in a variety of real-world tasks.

CHAPTER 4 : Predicting the future with transformational states

4.1. Introduction

Humans and other animals are able to reason about the future state of the world given visual observations of the present. Even as infants, humans can use images to make informed predictions of how objects and agents will move and act in the future (Spelke et al. 1996). A large body of evidence from the neural and cognitive sciences suggests that humans build predictive models of the world and use the resulting predictions to guide action and to learn better ways of engaging with the world (Bubic et al. 2010). The world is filled with image sequences, and it is clear that intelligent agents might use the rich dynamics of visual stimuli to guide learning. But how agents should learn to predict effectively remains an important computational problem.

The focus of this paper is the prediction of future images given a sequence of past images. Image prediction offers a general approach to tackling the challenge of state prediction in vision because it is not tied to a specific task or representation. By predicting images instead of task-dependent representations such as labels or segmentations, the agent gains more flexibility in how it uses information about the future. From this perspective, image prediction offers a unique opportunity for unsupervised visual representation learning, as image-level predictions can be used as a learning signal even in the absence of well-defined tasks or task-conditional reward signals.

Motion prediction is at the heart of future prediction in temporally contiguous video. Over short periods of time, scenes in the real world contain a slowly changing context and set of objects. Future frames can largely be predicted by modeling the motion of objects and the scene: that is, by transforming the current state into future states. Our work is motivated by the observation that learning states that can be transformed to produce future states may lead to representations that are easier to predict, as illustrated in Figure 13(A). Other methods that use motion for prediction typically rely on the assumption that image transformations can be modeled with local, piecewise translational motion. However, such methods struggle with scenes containing flexible objects like human bodies (e.g. the one shown in Figure 13(A)), due to the self-occlusion and non-rigid deformations that

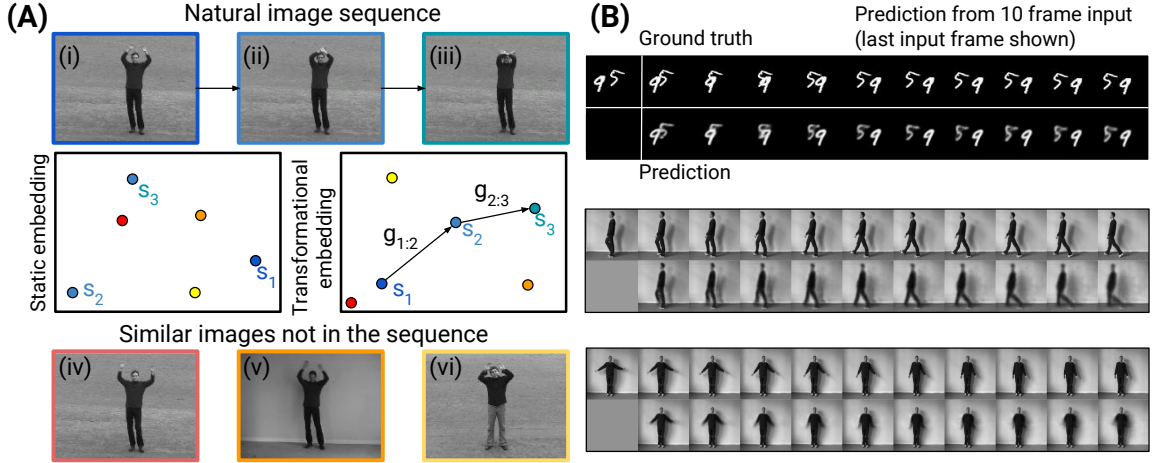


Figure 13: (A) This work is motivated by the observation that image transformations may be more easily modeled by a network that learns to transform latent states rather than transform or associate pixel intensities. By learning to model states, s , along with transformations between states, g , the RNN is encouraged to model sequences not by memorizing arbitrary transitions between certain images (static embedding) but by reshaping the embedding so that natural state transformations are predictable (transformational embedding). Figure best viewed in color. (B) Sample predictions of our model on sequences from the Moving MNIST and KTH datasets. We show only the last of 10 input images for visualization purposes. Our model produces good image predictions using only pixel-wise reconstruction losses.

such objects introduce. Here, we propose a model that makes predictions by transforming latent representations, and which can reason about transformations that are more complex than simple pixel translations.

To predict realistic images, a model of sequence transformations must also capture the appearance and texture of the scene as it transforms. This includes the content of image regions that appear or become dis-occluded over time. A model must capture the appearance of the foreground and background to paint in details of image regions that are revealed as the objects and scene move. A useful future image prediction model needs to model both this static state and its transformations to ensure that individual frames are realistic and that objects and the scene move realistically. Here, we show how to integrate weighted residual connections into our network to produce good models of background texture and other static scene content.

We propose to predict a latent state representation that encodes both the current state of the scene

and its transformation and that can be decoded to produce future images. Our method learns representations of states and transformations that are stable and sufficiently rich to produce multiple future frames without re-encoding estimated frames or repeatedly copying pixels from the input sequence. Our architecture learns to capture naturalistic motion in a variety of settings (synthetic and real) with minimal modifications. Our model achieves quantitative results competitive with the state of the art without assuming a static background (or stabilized preprocessing), without being constrained to directly copying or translating pixels from input frames, and without adversarial learning.

Our technical contributions are as follows:

- a novel RNN core formulation with a partitioned representation of latent states and transformations;
- a weighted, temporal residual connection that enables stably reconciling features across multiple time steps without re-encoding images;
- an encoder-decoder architecture that can be stably trained for good end-to-end image prediction without an adversarial loss.

4.2. Related work

There is a growing interest in predicting future imagery conditioned on past observations. This body of work leverages large, unlabeled video datasets to learn to make predictions. Prior work has explored a variety of aspects of the problem. Here, we present an overview of prior work organized by the level of abstraction of the target output, the generative process, the structure of the latent representation, and the loss function guiding learning. To further aid interpretation of the present work, we compare the specific design choices of a range of recently proposed models in Table 7 of the supplement.

Prediction targets At the prediction output level, a variety of representational levels have been targeted. At the highest level, several works have considered predicting semantic segmentation of

frames (Luc et al. 2017), deep visual image representations of frames (Vondrick et al. 2016b), human pose (Fragkiadaki et al. 2015; Bütepage et al. 2017; Martinez et al. 2017) and human actions (Nguyen and la Torre 2014; Kitani et al. 2012). Others have considered mid-level representation outputs, such as optical flow (Yuen and Torralba 2010; Walker et al. 2015). At the lowest and most general level, a growing body of research has explored predicting the pixel intensities of future frames (Ranzato et al. 2014; Srivastava et al. 2015a; Mathieu et al. 2016; Vondrick et al. 2016a; Villegas et al. 2017a; Vondrick and Torralba 2017; Liu et al. 2017; Kalchbrenner et al. 2017; Liang et al. 2017; Walker et al. 2017; Lotter et al. 2017; Cricri et al. 2016). In this paper, we propose a method to predict frame-wise pixel intensities by recurrently transforming image representations into the future.

Prediction and transformation A key differentiating aspect between prior work is the generative process. Inspired by encoder-decoder language models (e.g. Sutskever et al. 2014), several works consider a recurrent network that encodes an input sequence into a fixed length vector and a subsequent recurrent network that decodes the vector to progressively generate predicted frames (Ranzato et al. 2014; Srivastava et al. 2015a). Others have considered a more direct approach that predicts future frames from observed frames using a convolutional neural network (CNN) (Mathieu et al. 2016; Luc et al. 2017). Several other works have proposed copying or applying simple transformations to past pixels to generate image frame predictions (Brabandere et al. 2016; Finn et al. 2016; Liu et al. 2017; Vondrick and Torralba 2017; Pătrăucean et al. 2016). In contrast, we predict future images by transforming and decoding the latent space, rather than by directly predicting future frames or by copying or transforming past pixels.

Factored representations Another line of recent work has approached the problem of prediction by factoring the representation of the latent information or shaping the latent to learn properties useful for prediction. Vondrick et al. (Vondrick et al. 2016a) factor the generative process into separate foreground and background streams that are combined to create the final video. Goroshin et al. (Goroshin et al. 2015) train a linearized latent space so that future prediction can be treated as linear interpolation. Several works (Walker et al. 2017; Villegas et al. 2017b) have considered predicting human pose and then conditioning image generation on the predicted pose. These models

can achieve impressive results, but they assume latents with known structure (i.e. 2D poses) and are thus limited to human-focused imagery. Our method assumes only that learned representations can be transformed by a learned operator and is not restricted to settings where the latent space can be explicitly labeled.

Predicting with motion and content Most similar to our work are two approaches that factor the latent information to capture scene appearance and dynamics (Denton and Birodkar 2017; Villegas et al. 2017a). Denton and Birodkar 2017 learns separate representations of content and pose and predict future frames by fixing the content and estimating the future pose. This model produces very stable predictions, but it assumes content does not change over a sequence. This limits its applicability to scenes with camera motion and dynamic content. Our method does not assume fixed scenes, but instead learns a representation of state and motion that is designed to accommodate a variety of transformations while still preserving image structure.

Villegas et al. 2017a learns representations of content and motion by feeding two networks with images and difference images, respectively. They train their method with an adversarial loss and need to re-encode predictions to generate more than one future frame. While their method incorporates motion into the representation by splitting the input into images and difference images and directly predicting next frames, our method learns to represent both states and transformations from frames and learns motion by applying transformations to states. Our method produces good results without adversarial training or image re-encoding.

Loss functions Previous work has proposed to improve future prediction by designing loss functions to guide learning to better solutions. While earlier work uses standard pixel-wise reconstruction losses like the mean-squared error (MSE) or binary cross entropy (BCE) loss function (Srivastava et al. 2015a), more recent work (e.g., Mathieu et al. 2016; Vondrick and Torralba 2017; Lu et al. 2017; Zeng et al. 2017) often incorporates some form of generative adversarial network (GAN) model (Goodfellow et al. 2014), either alone or in conjunction with a reconstruction loss, such as MSE. While GANs can produce crisp predictions, they are notoriously hard to train and model convergence is difficult to evaluate (Goodfellow 2017). In this paper, we demonstrate that a simple MSE loss is

capable of generating good predictions when paired with an appropriately structured architecture.

4.3. Approach

In future prediction, we are given a sequence of T images $\{I_1, \dots, I_T\}$ and want to produce the most likely sequence of K future images $\{I_{T+1}, \dots, I_{T+K}\}$. We seek to do so by capturing how the structure of the image transforms over time.

Images are high dimensional but the pixel space dimension does not reflect the intrinsic dimensionality of the scene. For example, a 64×64 image of two translating digits lives in a pixel space of the same dimensionality as a 64×64 image of a walking person. However, the latter image contains scene content with many more degrees of freedom so its intrinsic dimensionality is higher. Similarly, a 128×128 and a 64×64 image of the same walking person both depict content with the same degrees of freedom (up to appearance details lost in downsampling), but with very different pixel dimensions. When we predict images, we must predict pixels, but we seek to do so by modeling the transformation of the images' content.

Accordingly, we model the instantaneous state of the scene at time t using a latent variable s_t . Because we do not know the state of future frames, we seek to transform past latent variables $\{s_1, \dots, s_T\}$ to estimate the future latents $\{s_{T+1}, \dots, s_{T+K}\}$. Future latents depend on previously estimated future latents, so we model this estimation process with a function f , such that the estimate at time $T+k$, where $1 \leq k \leq K$, is given by

$$\hat{s}_{T+k} = f(\{s_1, \dots, s_T, \hat{s}_{T+1}, \dots, \hat{s}_{T+k-1}\}). \quad (4.1)$$

In the context of image prediction, such a function is typically parameterized with a recurrent neural network (RNN) applied to the output of the encoder of an encoder-decoder architecture (Srivastava et al. 2015a):

$$\hat{I}_{T+k} = \text{CNN}_d(\hat{s}_{T+k}) \quad (4.2)$$

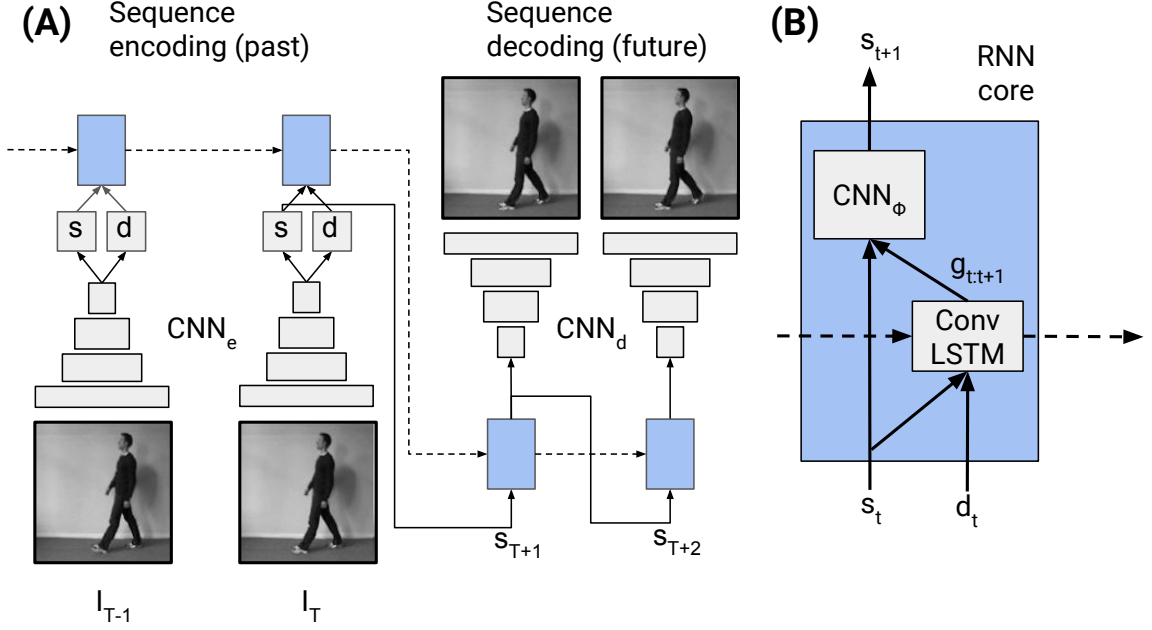


Figure 14: Architecture overview. (A) Our model uses an encoder-decoder sequence-to-sequence architecture with a factorized latent that captures the image state, s , and transformation, d . Residual connections are omitted for clarity; see the text and Figure 15 for details. (B) Future states are transformed from past states using an RNN core that accumulates the transformation estimate g with a ConvLSTM and applies it to the recursively estimated state s with an operator CNN_ϕ .

$$\hat{s}_{T+k} = \text{RNN}(\{s_1, \dots, s_T, \hat{s}_{T+1}, \dots, \hat{s}_{T+k-1}\}) \quad (4.3)$$

$$s_t = \text{CNN}_e(I_t), \quad (4.4)$$

where \hat{s}_t is the estimated latent state at time t , \hat{I}_t is the estimated image at time t , and CNN_e and CNN_d are encoder and decoder CNNs, respectively. This is illustrated in Figure 14(A).

While such structures can in principle learn to model arbitrary transformations (Siegelmann and Sontag 1995), these models often struggle to produce realistic image transformations. In practice, these models may learn to memorize transformations as arbitrary mappings from state to state (as illustrated schematically in Figure 13(A)) rather than representing transformations as predictable, generalizable mappings like those that characterize the natural transformations between world states.

We now describe how we encourage the network’s latent state to learn more predictable mappings by jointly learning representations of state and transformations.

4.3.1. Transformational states

To encourage an encoder-decoder structure to learn to model predictable latent space transformations, we introduce an additional latent variable d_t to capture the evidence available for estimating the transformation from each input image. The output of the encoder CNN at each time step can then be written as

$$s_t, d_t = \text{CNN}_e(I_t). \quad (4.5)$$

Next, we describe how we encourage the network to exploit the factorization in (4.5) by wiring the network so that transformational states d_t cannot directly predict images but must act by transforming states s_t .

We want the d_t to capture all information that is available from I_t about the transformation from state s_t to s_{t+1} . Let us call this transformation $g_{t:t+1}$. While individual images provide some information about how the world will move, in general they are insufficient to model the full transformation from t to $t + 1$ and to later points in the future.

To see this, consider the person in image (i) of Figure 13(A). From this world state, transformations in time are unlikely to produce image (v), which shows the same person in a different scene, or image (vi), which shows a different person in a similar position. However, the person in image (i) might move his arms closer together (producing image (ii)) or further apart (producing image (iv)). Given image (i) and (ii), however, image (iii) becomes much more likely than image (iv).

That is, the transformation that can be estimated from a single image (d_t) will not in general be equivalent to the true state transformation ($g_{t:t+1}$). However, some information about the transformational state of the world is observable from a single image, and we can arrive at better estimates by integrating transformation estimates over time. Accordingly, we use an RNN to incorporate the history of instantaneous transformation estimates d_t and the accompanying states s_t to obtain a better

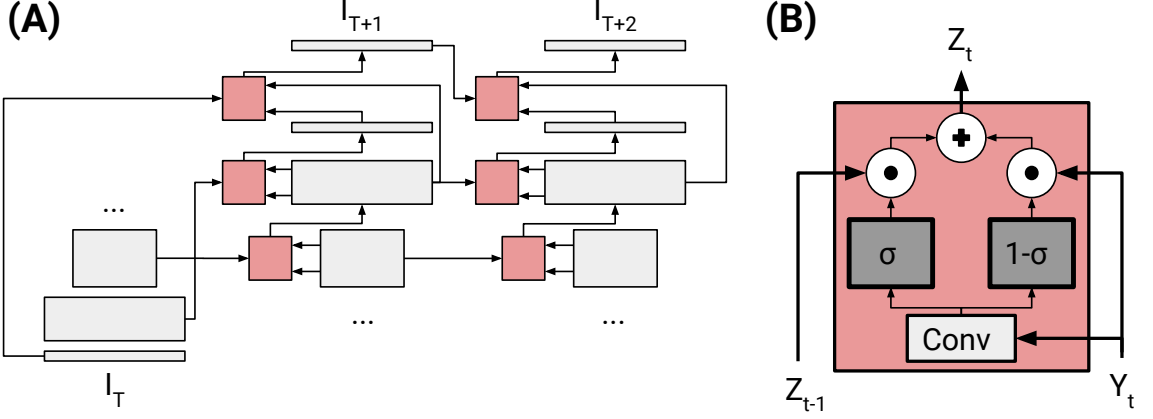


Figure 15: Weighted residual connections. (A) To produce high quality images at multiple time steps in the future without re-encoding images, we use a residual connection scheme designed to gradually alter image content from the last observed input image. Residual connections connect the encoder at time T (last input) to the decoder at time $T + 1$ (first output). At subsequent times, the decoder inherits information about the past from the decoder at the previous time only. The network has this connectivity pattern at every layer: we show only two decoder layers and the output image for easier visualization. (B) We use a retinotopic weighting scheme to allow each layer of a decoding network to selectively incorporate skipped input from the past. Weights and feature maps at time t are functions of the predicted latent state \hat{s}_t at time t .

estimate of the transformation $g_{t:t+1}$ acting on s_t :

$$g_{t:t+1} = \text{RNN}(\{[d_1, s_1], \dots, [d_t, s_t]\}). \quad (4.6)$$

We then model the action of this transformation on latent states as

$$s_{t+1} = \Phi(g_{t:t+1}, s_t), \quad (4.7)$$

where Φ is an operator that transforms s_t by applying $g_{t:t+1}$. We parameterize Φ with a three-layer CNN (with no recurrence), and we parameterize the RNN with a three-layer convolutional long short-term memory (convLSTM) model. We show the full recurrent core, including the CNN operator and transformation RNN in Figure 14(B).

We next describe how we integrate skip connections into the model to encourage long-term stability and fidelity of image production while the state is undergoing transformations.

4.3.2. Weighted residual connections

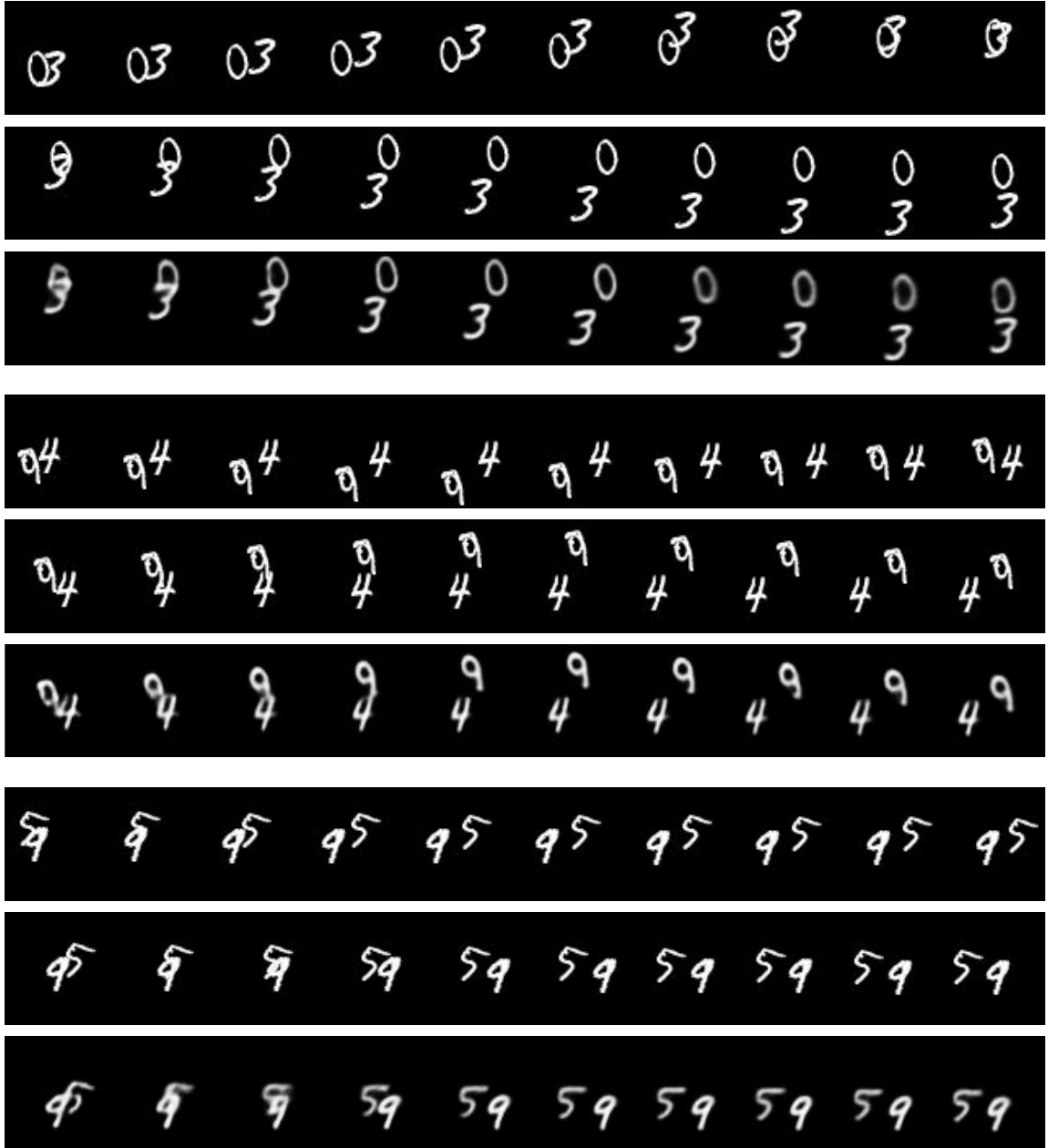


Figure 16: Example sequences on Moving MNIST. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. Our model is able to stably predict digits over multiple timesteps, even when digits overlap for multiple frames.

Recent works (Denton and Birodkar 2017; Villegas et al. 2017a; Finn et al. 2016) have found that

skip connections from encoder to decoder networks are essential for producing high quality image outputs, especially for capturing high-frequency information of textures and background. However, when encoded images are in the past and decoded images are in the future, this paradigm is limited in several ways. First, future encoder states cannot be used as a source of skipped image information without first re-encoding estimated images. This may lead to difficulties in CNN/RNN training because of mismatched statistics between true and estimated frames. Second, skipping from past states to future ones can introduce artifacts when static features are copied as if nothing in the scene has changed. This can result in ghosting artifacts that are difficult for the network to learn to correct.

We introduce a mechanism for passing information forward from the encoder state at the last input time step to the decoder at future time steps without re-encoding predictions and without repeatedly copying activations from the past (Figure 15). Instead of copying activations from the encoder to the decoder at all future times, we connect the encoder at the last input time step only to the decoder at the first prediction time step. Subsequent decoder time steps take the activations of the decoder at the previous time step and re-weight them. This configuration allows features to flow forward in time from the last input time step, while allowing features to change as necessary to reflect motion and without requiring images to be re-encoded.

The initial feature map Y_t^l output by layer l of the decoder network at time t is combined with skipped output Z_{t-1}^l from the previous time step in the form of a weighted residual connection:

$$Z_t^l = (1 - \sigma(W^l)) \odot Y_t^l + \sigma(W) \odot Z_{t-1}^l, \quad (4.8)$$

where \odot denotes element-wise multiplication. Z_t^l is the final output of the network at layer l at time t . W_t^l (a weight map) is the output of a 1×1 convolution with Y_t^l as input. We output one weight value for each spatial position of the feature map and broadcast the weight to all channels to perform the elementwise multiplication. This weighting strategy introduces only $K^l + 1$ parameters per layer, where K^l is the number of channels in Y^l .

For the first prediction time step, the skip inputs Z_{t-1}^l come from the layer of the encoder network

Model	average, 10 predicted frames	first frame prediction
ConvLSTM (Shi et al. 2015)	367.2	-
Encoder-Decoder LSTM (Srivastava et al. 2015a)	341.2	-
Dynamic Filter Networks (Brabandere et al. 2016)	285.2	-
Spatiotemporal Autoencoder (Pătrăucean et al. 2016)	-	179.8
Video Pixel Networks (Kalchbrenner et al. 2017)	87.6	-
Video Ladder Networks (Cricri et al. 2016)	187.7	-
Ours	210.1	172.4

Table 4: Comparison of binary cross-entropy (BCE) results (nats/frame) on the Moving MNIST test set. Lower scores indicate better performance.

at the last input time step with matching spatial dimension. Otherwise, they come from the corresponding layer in the decoder at time $t - 1$. The weighting scheme is shown in Figure 15(A). This configuration is similar to the one introduced in Srivastava et al. 2015b, applied at each time step.

When paired with our network architecture, this skip configuration allows us to estimate future images without re-encoding estimated images into the encoder CNN. Because subsequent time steps inherit the activations of the previous decoder state, and do not directly copy the states of the last encoder (as in e.g. Denton and Fergus 2018), we observed that these networks trained more quickly and resulted in fewer ghosting artifacts.

We incorporate a similar weighted residual scheme to directly skip the last input image to future timesteps. As with feature maps, for all times $t > T + 1$ we skip the image from the previous timestep $t - 1$ instead of the last input image I_T . Directly skipping the final input image to later time steps resulted in lower quality outputs (see Figures 26 and 27 in the supplemental material for examples). We also observed that the residual connection works best when the weighting is applied after the tanh nonlinearity in both images. Weighting before the output nonlinearity led to saturated images at later prediction time steps.

4.4. Experiments

4.4.1. Datasets

We performed experiments on three datasets: a standard synthetic dataset, Moving MNIST (Srivastava et al. 2015a), and two real world datasets, KTH actions (Schüldt et al. 2004) and UCF101 (Soomro

Model	PSNR		SSIM	
	at time 1	average over 10 frames	at time 1	average over 10 frames
ConvLSTM (Villegas et al. 2017a)	33.8	27.6	0.95	0.84
MCNet (Villegas et al. 2017a)	33.8	28.2	0.95	0.86
Ours	34.8	29	0.95	0.86

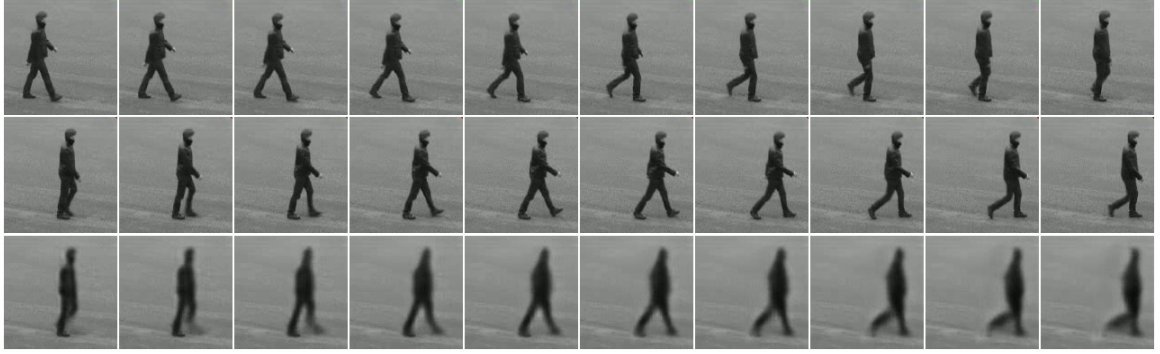
Table 5: Comparison of frame prediction results on the KTH test set. Higher scores indicate better performance.

Model	PSNR	SSIM
EpicFlow (Revaud et al. 2015)	29.1	0.91
NextFlow (Sedaghat et al. 2017)	29.9	-
BeyondMSE (Mathieu et al. 2016)	28.2	0.89
DVF (Liu et al. 2017)	29.6	0.92
Dual Motion GAN (Liang et al. 2017)	30.5	0.94
Ours	28.3	0.88

Table 6: Comparison of next frame prediction results on the UCF101 test set (split 1). Higher scores indicate better performance.

et al. 2012). Moving MNIST is a dataset of synthetic videos, with an arbitrarily large training set (videos are generated procedurally) and a test set of 10,000 videos. Each video has an image resolution of 64×64 and is 20 frames in length. The videos capture two digits moving in random directions and with random velocities. KTH consists of 2391 videos capturing six human actions: boxing, hand clapping, hand waving, jogging, running, and walking. As is standard practice in prior work on frame prediction using KTH, we convert the images to 128×128 prior to processing. All sequences contain scenes with relatively homogeneous backgrounds. The scenes were captured with a static camera at 25 frames per second. UCF101 contains 13,320 YouTube videos capturing 101 human actions. As done in previous evaluations using UCF101, we convert the images to 256×256 prior to processing. Notably, many UCF101 videos contain spatial and temporal (i.e. duplicate frames) artifacts due to compression.

4.4.2. Architecture and training details



(a) Walking



(b) Running



(c) Hand waving

Figure 17: Example sequences on KTH. For all three examples, the first row shows the input sequence (past), the second row shows the ground truth future, and the third row shows the predicted sequence. The model produces faithful motion in a variety of settings and is able to paint in the background after dis-occlusion.

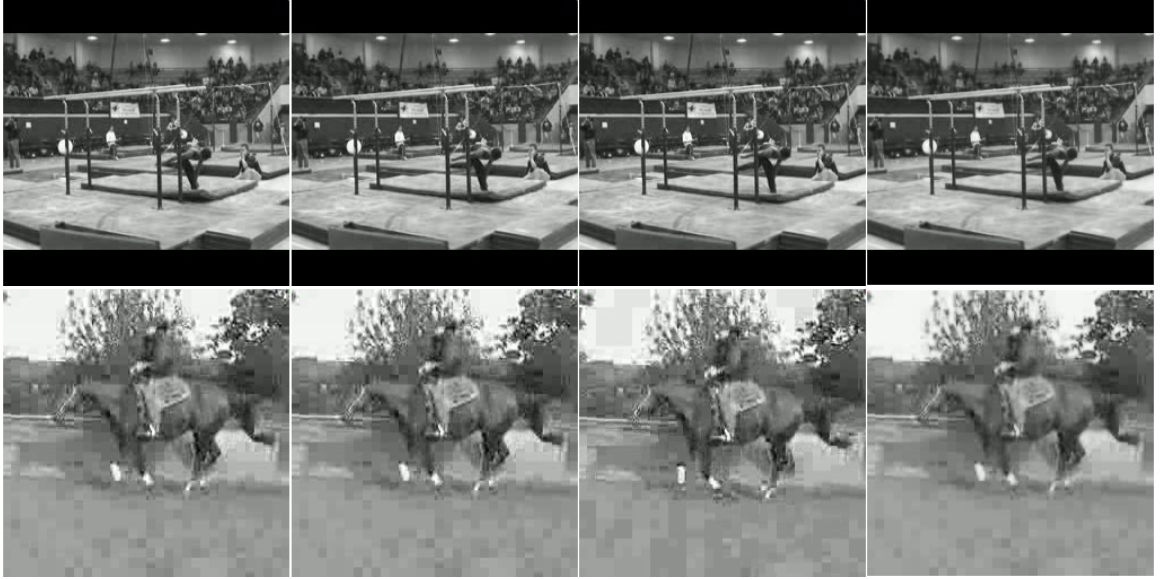


Figure 18: Example sequences on UCF101. For each example, we show two frames from the past followed by the ground truth third frame and the third frame predicted by the model from the first two images.

The Moving MNIST and KTH networks were trained to predict 10 frames given 10 input frames and UCF101 networks were trained to predict 1 frame given 2 input frames (to allow us to compare to the compendium of state-of-the-art results in Liang et al. 2017). On all datasets probed, we trained the network end-to-end using an average pixel-wise reconstruction loss between the estimated sequences and ground truth future sequences. We use an MSE loss for KTH and UCF101 and a BCE loss for Moving MNIST. All networks were trained using stochastic gradient descent (SGD) with momentum. We used a starting learning rate of 1 on KTH and UCF101 and 10 on Moving MNIST. We decayed learning rates by a factor of 10 every time the validation loss reached a plateau, until convergence. We used a momentum value of $\beta = 0.5$ in all cases. We used a weight decay of 1×10^{-4} for encoder and decoder weights on all networks, and we included dropout with a rate of 0.5 in all hidden layers of encoder networks on UCF101 and KTH.

We used horizontal mirroring and random cropping for data augmentation on both UCF101 and KTH datasets. We trained on Moving MNIST with 50 sequences per batch, on KTH with 20 sequences per batch, and on UCF101 with 10 sequences per batch.

The convolutional architectures used on all three datasets are based on the DCGAN architecture (Radford et al. 2016). Each layer of the decoder except for the input layer contains the same number of channels as the corresponding layer of the encoder architecture. Because the decoder does not take the transformational latent as input, the decoder input is of size $4 \times 4 \times N_s$, while the encoder output is of size $4 \times 4 \times (N_s + N_d)$, where N_s is the number of channels in the state latent s and N_d is the number of channels in the transformational latent d . In all architectures used here, $N_s = N_d$. We did not perform hyperparameter search for the values of N_s and N_d or the architectures used for encoders and decoder CNNs, and it is likely that better results can be obtained using optimized settings.

The architectures we use on Moving MNIST, KTH, and UCF101 differ only in the number of layers and the number of filters per layer in the encoder and decoder CNNs. Architecture depths were chosen so that the spatial size the encoder output (and decoder input) was 4×4 . We specify full architectures in the supplemental material (supplemental section C.2). We will make the model code and trained models available upon paper acceptance.

4.4.3. Evaluation

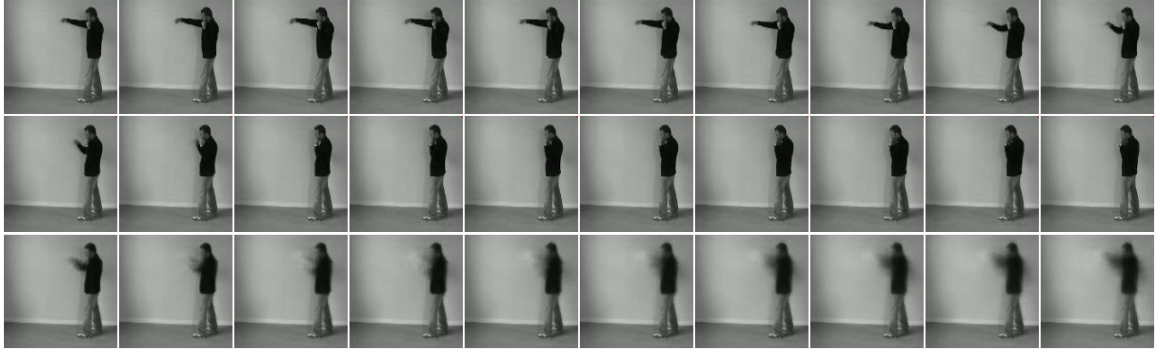
It is difficult to quantitatively evaluate prediction results because reconstruction errors and other measures do not generally fully capture the perceptual quality of reconstructed images (Wang et al. 2004; Theis et al. 2016; Mathieu et al. 2016). Nonetheless, quantitative evaluations can give a reasonable indication of the average quality of a method when seen alongside the qualitative results the method produces.

We evaluate our methods using the error measures most commonly used in the literature: binary cross entropy for Moving MNIST (Srivastava et al. 2015a) and peak signal to noise ratio (PSNR) and Structural Similarity (SSIM) (Wang et al. 2004) for KTH and UCF101. We evaluate SSIM using a window of 7x7 pixels with uniform weighting (the same parameters as Denton and Birodkar 2017).

We show quantitative results for Moving MNIST in Table 4, KTH in Table 5, and UCF in Table 6. In all cases, our results are competitive with state of the art. Because of the large number of architectures and loss configurations in the literature, it is infeasible to thoroughly test all architecture and loss

configurations. We report results based on the numbers used in the literature. To aid interpretation of our results in the context of the sequence prediction literature, we include a table comparing the different architectural and loss configurations in the supplement (Table 7).

We show sample qualitative results on the three datasets in Figure 16 (Moving MNIST), Figure 17 (KTH), and Figure 18 (UCF). Our method produces reasonable results with good motion in many of settings in these three datasets. The output of dynamic models are difficult to evaluate based on static images alone, and consequently the results of our method are best understood by examining the videos on the project website (https://daniilidis-group.github.io/transformational_states). To aid interpretation of our results, we also show failure cases on KTH in Figure 19. Additionally, we show prediction results produced by models with network ablations in the supplement: ablations on Moving MNIST are shown in Figure 25 and ablations on KTH are shown in Figures 26 and 27. Ablation results are shown on random sequences from the test data in all cases for fair comparison.



(a) Boxing



(b) Running



(c) Walking

Figure 19: Example failure cases on KTH. (a) The model outputs a blurry motion sequence that does not correspond to the ground truth. (b) The model fails to correctly predict motion or paint in the background when the moving object occupies only a small part of the image. (c) The model fails to correctly paint in the background after the foreground moves, leading to ghosting artifacts.

4.5. Conclusion

We have described a model for predicting sequences of future images using an architecture that learns latent states and their transformations to future states. We show how to couple this architecture

with weighted residual connections from past to future time steps to produce images that are stable after recursive transformations. The resulting network can be trained to predict reasonable results on synthetic and real datasets without requiring direct pixel copying or a GAN. Our model produces good qualitative results and achieves quantitative results comparable to state-of-the-art on several image prediction datasets.

CHAPTER 5 : Conclusion

5.1. Future work

In this dissertation, I have presented several lines of work developing the use and understanding of global motion. There are still many challenges to and limitations of our current understanding of motion, especially in regard to how motion can be used for prediction, for building invariant representations, and as the basis for control strategies. In this section, I present several promising directions in which the work presented in this dissertation can be extended. I divide this section into four parts, exploring (i) how motion and prediction might improve our understanding of the function of visual cortex, (ii) how better models for long-term prediction might be developed and their relevance for control and reinforcement learning (iii) how unsupervised approaches to motion might be used for control by learning the degrees of freedom of objects, and (iv) how models of motion and prediction might be used to develop notions of the invariances and symmetries of objects for better imitation learning.

5.1.1. Developing functional DNN models of the dorsal stream

DNNs have proven to be very fruitful systems for accounting for the responses of populations of neurons in a wide variety of systems in the past years. While there has been great success explaining the responses of neurons in ventral areas (Yamins et al. 2014; Cadieu et al. 2014; Cadena et al. 2017), less progress has been made in accounting for dorsal responses. One recent study (Tacchetti et al. 2017) showed that a DNN trained for human action recognition could be used to produce response patterns similar to those obtained by magnetoencephalography (MEG) recordings of humans performing the same task. While these results are suggestive, they are limited because of the lack of location specificity of the recordings and because the models were not trained to good performance on a real-world dataset, so it is difficult to understand how well these results will predict responses in a wider variety of settings. More generally, action recognition may not be a general enough task to probe the full representation of motion in the dorsal stream, as it is possible for action recognition models to achieve good performance with only limited reliance on temporal features (e.g. consider

the purely spatial baselines used in Simonyan and Zissermann 2014).

Visual prediction models may serve as alternative functional models of dorsal stream computation. As discussed in Chapters 1 and 4, successful prediction relies on good modeling of the static and dynamic components of the visual scene and on the global representation of motion. This suggestion is buttressed by recent work (Watanabe et al. 2018) showing that a visual prediction model (Villegas et al. 2017a) can reproduce the phenomenology of a human visual motion illusion. This suggests that visual prediction models may indeed learn representations like those underlying visual motion processing in the brain. Moreover, several works have shown that visual prediction can serve as a good functional model of neural responses in other motion-related visual systems, such as mouse V1 (Leinweber et al. 2017) and the fly retina (Palmer et al. 2015). As successful visual prediction relies on many of the computational mechanisms needed for spatiotemporal perception, it is a very promising area to search for analogues of dorsal stream computation.

One of the long-term goals of visual neuroscience is to explain the functional role of the combined dorsal-ventral system. Any such account must reproduce the properties of ventral stream computation, including their role in producing disentangled representations of object identity (DiCarlo et al. 2012). The tools of visual prediction offers a route to exploring how the dorsal and ventral systems interact. We have so far focused our discussion of visual prediction on image prediction, but an emerging body of work in computer vision has explored how similar models can be used to predict the future state of visual representations (Yuen and Torralba 2010; Vondrick et al. 2016b; Luc et al. 2017; Luc et al. 2018). Notably, Vondrick et al. 2016b proposes a model that can predict the output of another model trained for object recognition at future time steps. Accordingly, this model can be seen as generalizing feedforward object recognition models to the spatiotemporal domain.

Similarly, the methods developed in chapter 4 of this dissertation can be naturally extended to predict transformations of task-directed latents rather than latents for image prediction. Such a model would be well suited for producing representations that capture both object representation features (because the output at each timestep would have a similar form to the representations produced by the models used by e.g. Yamins et al. 2014 to probe ventral areas) and dynamic features (because the

development of the representation in time would be subject to the same functional requirements as predictive models of images). Models of this form may have the potential to simultaneously account for dorsal and ventral activity.

5.1.2. Long-term prediction for control

One of the primary goals of the perception of dynamic content is to develop representations that are useful for learning how to act and control in the real world. As a perceptual strategy, prediction is particularly well suited for this type of goal (Littman et al. 2002). Several groups have recently developed strategies that directly integrate predictive perception into control models (Dosovitskiy and Koltun 2017; Jaderberg et al. 2017; Wayne et al. 2018), that use prediction to facilitate task learning (Finn et al. 2016; Byravan et al. 2017), or that use prediction to simulate world dynamics to allow learning in simulation (Oh et al. 2015; Chiappa et al. 2017; Ha and Schmidhuber 2018).

As noted in Chapter 4, one major challenge for prediction models is the generation of realistic predictions long into the future. Prediction is useful in control and reinforcement learning contexts largely because it can be used to learn representations that capture the effects that actions can have on the environment. Indeed, the most general of the recent wave of models using prediction for action (Wayne et al. 2018) succeeds largely because it is able to achieve better foresight by integrating a model of memory. A properly designed long-term prediction system can act as a complement to memory, allowing even more robust foresight to be learned. Accordingly, developing predictive models suited for long term prediction may help immensely in these domains.

The mechanisms for reconciling past and future information in predictive models developed in Chapter 4 offer one path towards developing longer term representations. We saw that properly structured residual connections between past and future states could lead to stable predictions that nonetheless changed appropriately to produce naturalistic motion. In future work, I plan to extend these models by incorporating insights from very deep hourglass networks used for progressive refinement of human pose estimates (Newell et al. 2016; Pavlakos et al. 2017).

Hourglass networks estimate pose by iteratively encoding and decoding pose estimates. At each

iteration, a different set of weights is used to perform the encoding and decoding, taking the previous estimate and the intermediate computations used to produce it as input. This configuration allows very long chains of computation, such as those needed to produce long-term estimates, to be trained efficiently. In these settings, skip and residual connections are used between stages of processing to encourage deeper, more powerful functions to be learned (He et al. 2016).

This strategy can be adapted to the problem of long-horizon prediction, with images at short time delays treated as intermediate estimates. As in the pose literature, this strategy may be able to encourage more stable final estimates that take information at all spatial and temporal scales into account. We are currently exploring incorporating this strategy with architectures designed to encourage motion representation. By developing models for better long-term prediction in this way, we may be able to endow agents with better foresight and encourage them to develop more intelligent, non-reactive behavior.

5.1.3. Discovering degrees of freedom

I have discussed the role that a good predictive model can play in improving performance in reinforcement learning settings. One reason for this is that prediction may implicitly capture good representations of the degrees of freedom (DoFs) of a system without requiring they be specified in advance. Modeling the DoFs of a scene is especially important for agents who must learn how to control end effectors without knowing in advance how to make use of them¹. A representation of DoFs that is independent of a hard-coded control readout is useful when robustness to changes in the readout is desired. This is the case in domains such as in autonomous driving. In this setting, a representation of car DoFs that are robust to changes in the steering controls of the car would facilitate generalization between cars of different models.

We are currently exploring methods taking advantage of this property to learn to represent the actions available to an agent without specifying the control interface (i.e. without telling the agent

¹Shimon Ullman has described a role for unsupervised learning of this type in human infants (Ullman 2016), and he describes evidence that suggests infants preferentially attend to others hands even before they have learned to manipulate objects themselves.

which controls correspond to which motor primitives). By coupling advances in DNN-based visual predictive techniques with strategies from the information bottleneck (IB) literature (Tishby et al. 1999) to encourage the representation to encode the changes in the scene but not the state information, we may be able to capture such a DoF representation. There is increasingly strong evidence that training using IB principles can lead to maximally disentangled representations (Tishby and Zaslavsky 2015; Achille and Soatto 2017; Saxe et al. 2018)², and hence to representations with clear representations of independent DoFs in this context. Recent work has shown that a model that learns to predict future images using past images and past controls can be repurposed to successfully perform simple visual servoing tasks (Finn et al. 2016). Such an approach requires annotation of actions for the training sequences, which limits its applicability and requires more costly data collection. A representation that combines IB principles with DNN-based prediction may facilitate control of an agent using representations trained just from observations of the agent and without requiring knowledge of the action space it uses.

5.1.4. Learning symmetries

A goal complimentary to that of learning the degrees of freedom of a scene is learning the invariances of the content of the scene - that is, learning its symmetries. A symmetry of an entity is a property of the entity that do not change when transformations are applied to it. For example, the shape of a rigid body (e.g. as parameterized by the coordinates of its points in a reference frame attached to the body) is a symmetry of the body under translation and rotation: translating and rotating the body will not change its shape. Similarly, the shape of a human body captures its symmetries under pose transformations: if a person moves their arms from their side to above their head, their shape and its properties (such as weight, height, arm length, etc.) do not change. Note that not all transformations preserve a given symmetry: gaining ten pounds over the course of a week does lead to changes in a human's shape. Symmetric properties like the shape of a human body and other nonrigid objects under re-posing cannot be easily captured in terms of simple transformations, and modeling them remains difficult. Current methods for learning object shape (e.g. Su et al. 2015; Qi et al. 2017;

²In particular, we are interested in representations that disentangle the DoFs controllable by the agent from those it cannot control, such as factors of variation in the environment or due to the actions of other agents.

Tulsiani et al. 2017) are notoriously data hungry, so more efficiently expressive methods will be essential in learning non-rigid objects.

The ability to model such symmetries is important for tasks that rely on relating the observed state of an entity to the range of states it can express. One such task is imitation learning. In imitation learning, one agent (the imitator) attempts to produce actions that mimic the actions produced by another agent (the demonstrator). Imitation learning is challenging even if the imitator and demonstrator have identical bodies, but the task is especially difficult when (i) the imitator and demonstrator have different bodies, and (ii) the goal of the action is not obvious from the action itself (Ho and Ermon 2016; Merel et al. 2017). There is evidence that humans can abstract goals from the specifics of behavior even as infants (Gergely et al. 2002), but it is not clear how to reproduce this behavior in artificial agents.

A good representation of the symmetries of the demonstrator’s body may facilitate an imitator’s attempts to build analogies between the demonstrator’s actions and its own, and hence facilitate learning the demonstrator’s goals. The relevant symmetries of nonrigid objects are typically expressed in changes made in time. For example, as a human body moves, its appearance changes in accordance with changes in pose, not by changing the shape: a person’s arms may move from their side to their head, but that person does not typically spontaneously gain twenty pounds. Methods developed to understand motion, such as those presented in this dissertation, may be useful for future work developed to learn about these symmetries more efficiently. In particular, the results presented in Chapter 3 suggest that the operations corresponding to change in the environment can be learned by enforcing group operations. Indeed, algebraic groups are the natural framework for describing the symmetries of objects, and developing strategies to represent these groups is a promising direction for future work.

For complex systems, such as those involving perception and action in real world environments, it is a non-trivial task to discover the form of those symmetries. Consider a non-rigid shape, such as a human body, under the transformations produced by changes in its pose. The body’s shape, i.e. as parameterized by the pose-independent connectivity structure of its surface mesh, is a symmetry

of the body. We can approximately capture this invariant shape by hand-designing factorizations of shape and pose and learning the components of these models from data, as in (Anguelov et al. 2005; Loper et al. 2015; Pons-Moll et al. 2015; Li et al. 2017). These models use learning to capture the invariance structure of bodies in a low-dimensional fashion, but the factorization into poses and shapes is designed and hard-coded. These representations must be extensively engineered and are limited in their application to objects with known structure, such as bodies or faces. In the long term, a framework to learn such symmetries directly from data would be immensely valuable to modeling generic objects and learning how they act.

In future work, I plan to develop rules for learning group structures in the service of other representational tasks with the goal of discovering such symmetries. The strategy described in Chapter 3 gives a framework for learning the group that captures the symmetries exhibited by an environment, but the learning rule as discussed is not powerful enough to capture reusable or general group descriptions. Future work will focus on improving its power by forcing the representation that is learned this way to be used on a task. To do so, the group operation should be treated as a group action on the set of states useful for some representational task.

In the context of image prediction, a scheme similar to the one described in Chapter 4 can be used. That approach learns a representation of motion, $g \in G$, that acts on a representation of state, s . By imposing a group structure on the set of motions G , this representation is encouraged to capture group properties while maintaining its interpretation as an action on the set of image states. This approach may be able to learn reusable representations of these symmetries by grounding the learned representations in well-defined tasks and by training on multiple visual domains.

5.2. Summary

In this dissertation, I have developed methods for learning and using global representations of motion. I have described methods to robustly estimate metric global properties of motion, to learn unsupervised representations of global motion, and to learn representations of motion to improve image sequence prediction. These methods address current challenges for the understanding and use

of visual motion, and they point the way forward to future work using motion to represent the global, nonrigid world in an actionable manner.

In the 1980 paper in which Kunihiro Fukushima proposed the Neocognitron, the antecedent of modern CNN models of visual computation, he motivated his work by noting that:

The mechanism of pattern recognition in the brain is little known, and it seems to be almost impossible to reveal it only by conventional physiological experiments. If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism of the brain. Fukushima 1980

The current overlapping state of neuroscience, computer vision, and machine learning is very exciting and has already led to significant advances in the set of tools we have for understanding visual computation. I believe the stage is set for dramatic progress in our understanding of how agents can learn and compute in the dynamic visual world and how these computations might be instantiated in the brain.

APPENDIX A : Supplemental material: Fast, robust, continuous monocular egomotion computation

A.1. Supplemental experiments

A.1.1. Goodness of fit for Laplacian distribution

To justify the use of a Laplacian distribution for ERL, we used ground truth flow fields to examine the distribution of errors in estimated optical flow. Ground truth flow was obtained using the KITTI Stereo dataset. Flow was produced according to the motion field equation (equation (2.1)). All images containing both depth and odometry ground truth were used. Errors were obtained for flow at all points that both contained ground truth depth and produced a sufficiently good KLT flow vector, using the same inclusion criteria as the main paper. We fit Laplacian and Gaussian distributions to the errors in the estimated optical flow, and computed the sum of the log likelihoods of each errors in the estimated distributions. In Figure 20 we plot the relative likelihoods of the data under the two distributions, and it is clear that the Laplacian fits consistently produce a higher likelihood than the Gaussian fits.

A.1.2. Comparison Of ERL likelihood schemes

We also compared the results obtained by ERL different likelihood functions over the KITTI Odometry dataset. We compare the results obtained using a Laplacian or a Gaussian fit to compute the weights in ERL. Results are shown in Figure 21. The use of a Laplacian distribution leads to a small but consistent improvement.

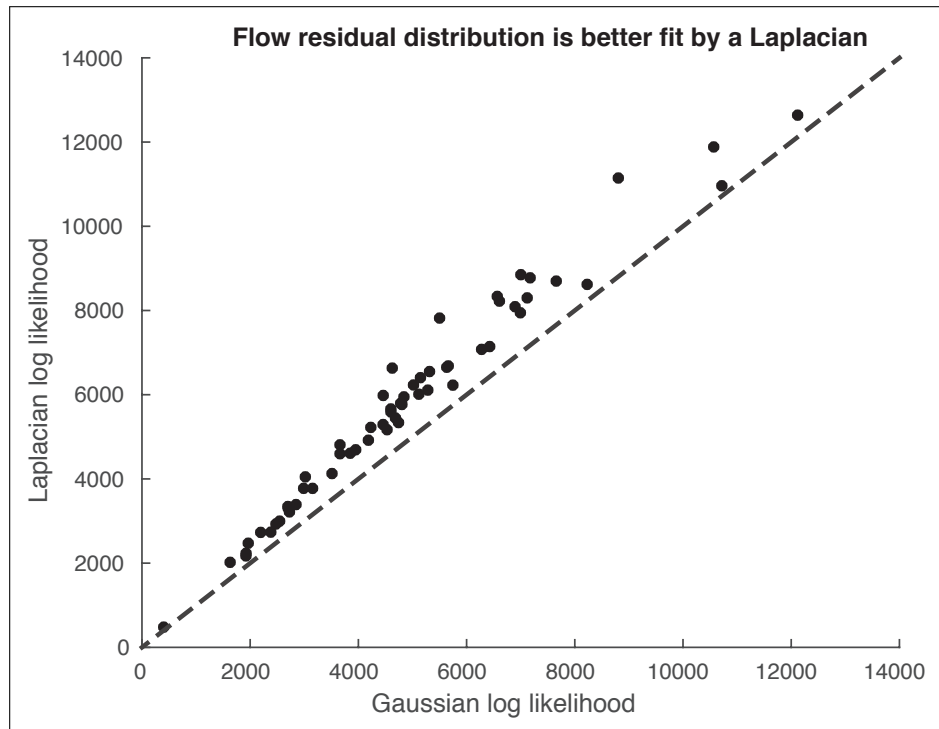


Figure 20: Log likelihoods of Laplacian and Gaussian fits to the optical flow error. Laplacian fits are consistently better than Gaussian fits.

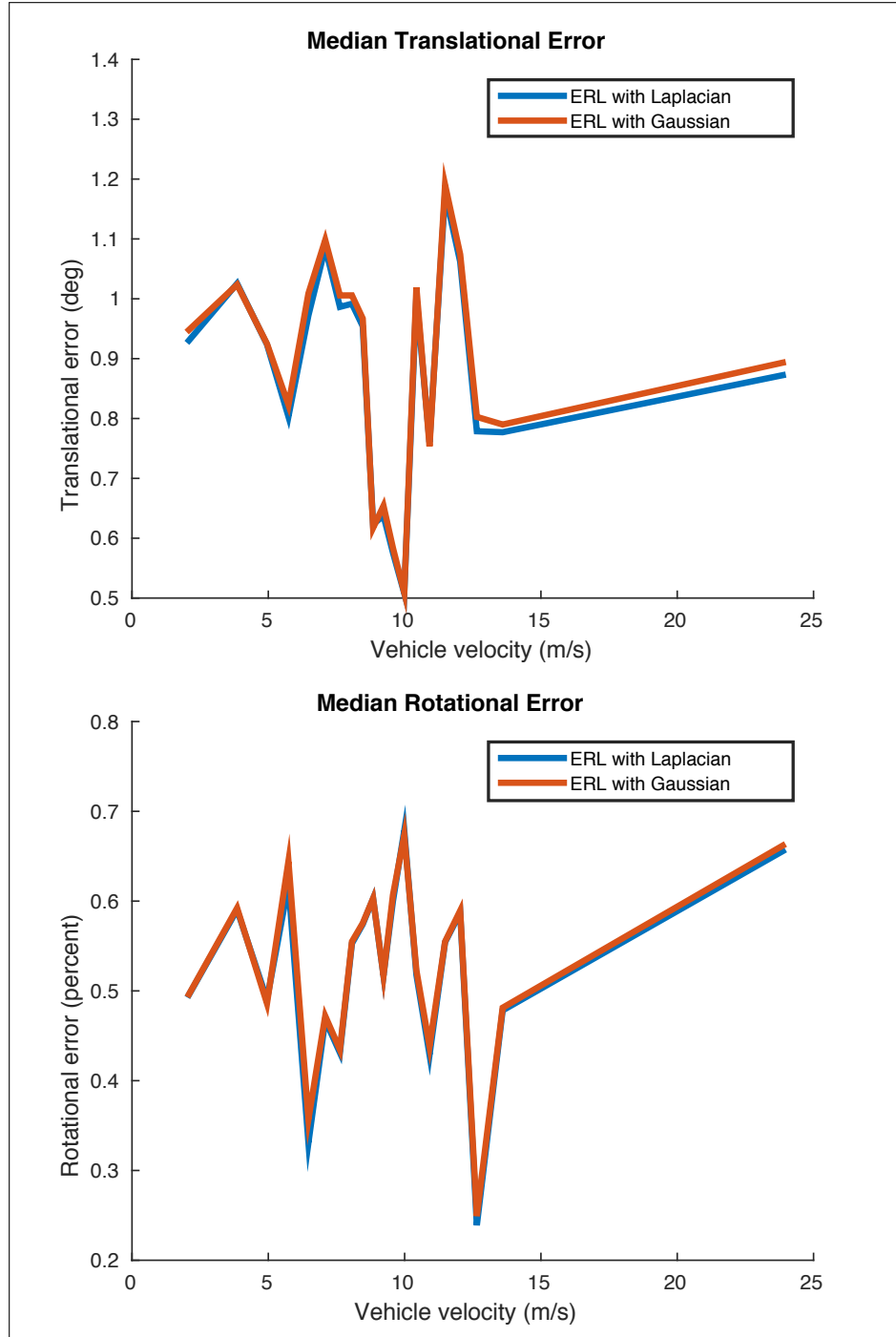


Figure 21: Median translational and rotational errors on the full KITTI odometry dataset for ERL with two candidate distributions.

A.2. Derivation of linear least squares estimate

We first minimize equation (2.10) from the main paper with respect to the inverse depths ρ , giving

$$\begin{aligned}
& \min_{\rho} E(t, \rho, \omega) \\
&= \min_{\rho} \|A(t)\rho + B\omega - u\|_2^2 \\
&= \left\| -A(t) \left(A^\top(t)A(t) \right)^{-1} A^\top(t)(B\omega - u) + B\omega - u \right\|_2^2 \\
&= \left\| \left(I - A(t) \left(A^\top(t)A(t) \right)^{-1} A^\top(t) \right) (B\omega - u) \right\|_2^2 \\
&= \|A^\perp(t)^\top (B\omega - u)\|_2^2.
\end{aligned}$$

We now have an expression in terms of the orthogonal complement of $A(t)$. Finding this orthogonal complement is fairly simple since it is sparse. To show this, we first note that the orthogonal complement of $A(t)$ is the null space of $A^\top(t)$. $A^\top(t)$ is of the form

$$A^\top(t) = \begin{bmatrix} t^\top A^\top(x_1) & 0 & \dots & 0 \\ 0 & t^\top A^\top(x_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & t^\top A^\top(x_n) \end{bmatrix} \in \mathbf{R}^{n \times 2n}.$$

Each of the rows of $A^\top(t)$ are orthogonal, so we can consider each of the rows individually. Consider the vector

$$\phi_i = \left(0, 0, \dots, 0, t^\top A^\top(x_i)J^\top, 0, \dots, 0 \right)^\top,$$

where

$$J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

is a skew-symmetric matrix in $\mathbf{R}^{2 \times 2}$. By construction, this vector is orthogonal to the i^{th} column of

$A^\top(t)$. We normalize and concatenate these vectors to form the matrix

$$A^\perp(t) = \begin{bmatrix} \frac{\phi_1}{\|\phi_1\|} & \frac{\phi_2}{\|\phi_2\|} & \cdots & \frac{\phi_n}{\|\phi_n\|} \end{bmatrix}.$$

This matrix is very sparse, so we can compute products with it very efficiently: $A^\perp(t)^\top B$ and $A^\perp(t)^\top u$ can be computed in $\mathcal{O}(n)$ time. From here, the least squares estimate of ω can be computed as:

$$\hat{\omega}(t) = \left(B^\top A^\perp(t) A^\perp(t)^\top B \right)^{-1} B^\top A^\perp(t) A^\perp(t)^\top u. \quad (\text{A.1})$$

In summation notation:

$$\hat{\omega}(t) = \left(\sum_{i=1}^n \frac{B^\top(x_i) J A(x_i) t t^\top A^\top(x_i) J^\top B(x_i)}{\|J A(x_i) V\|^2} \right)^{-1} \left(\sum_{i=1}^n \frac{B^\top(x_i) J A(x_i) t t^\top A^\top(x_i) J^\top u_i}{\|J A(x_i) t\|^2} \right). \quad (\text{A.2})$$

There are $2n$ terms to compute, and one inversion of a 3 by 3 matrix, making this $\mathcal{O}(n)$ time to compute. Taken altogether, we compute the residual given t by

$$\|A^\perp(t)^\top (B \hat{\omega}(t) - u)\|_2^2.$$

To compute the residual, we use the error vector E , defined as:

$$E_i(t) = \frac{t^\top A^\top(x_i) J^\top}{\|J A(x_i) t\|} (B(x_i) \hat{\omega}(t) - u_i) \quad \text{for } i = 1, \dots, n \quad (\text{A.3})$$

$$E(t) = (E_1(t), E_2(t), \dots, E_n(t))^\top. \quad (\text{A.4})$$

The residual is exactly $\|E(t)\|^2 = \sum_i \|E_i(t)\|^2$. As there are n of these error terms, and $\hat{\omega}$ takes $\mathcal{O}(n)$ to compute, this residual calculation takes $\mathcal{O}(n)$ to compute. This was shown to be an unbiased estimator in Zhang and Tomasi 1999.

A.3. Lifted weights formulation

Now, since the cost function is given as a sum of squares, we can optimize it using a Gauss-Newton framework. Therefore, to reject outliers we can use Zach 2014 to optimize this efficiently. Fixing the t term, the equation becomes linear:

$$\min_{\omega} \|A^\perp(t)^\top B\omega - A^\perp u\|_2^2 = \|f(\omega)\|_2^2.$$

So we see the Jacobian is given by $\nabla f(\omega) = A^\perp(t)^\top B$ and thus, as in Zach 2014, our lifted cost function takes the form

$$\min_{\omega, w} \left\| \begin{pmatrix} w \circ f(\omega) \\ \kappa(w \circ w) \end{pmatrix} \right\|_2^2.$$

Here we use the smooth truncated quadratic for our κ function (applied elementwise):

$$\kappa(w^2) = \frac{\tau}{\sqrt{2}}(w^2 - 1),$$

where τ is a hyperparameter. Therefore the Jacobian used for the Gauss-Newton iteration is:

$$\mathbf{J} = \begin{pmatrix} \text{diag}(w)\nabla f(\omega) & \text{diag}(f(\omega)) \\ \mathbf{0} & \nabla \kappa(w \circ w) \end{pmatrix}$$

where ∇f and $\nabla \kappa$ denote the Jacobian of f and κ , respectively. From here, we follow the derivation in Zach 2014.

A.4. Implementation details of Soatto/Brockett algorithm

A.4.1. Rotation estimation

Here, we derive an expression for $\hat{\omega}$. First, recall equation (2.4). We can rewrite this as:

$$\hat{\omega}(t) = G_{full}(t)^{-1} H_{full}(t), \tag{A.5}$$

where

$$G_{full}(t) = \sum_{i=1}^n \frac{B^\top(x_i)JA(x_i)tt^\top A^\top(x_i)J^\top B(x_i)}{\|JA(x_i)V\|^2} \quad (\text{A.6})$$

$$H_{full}(t) = \sum_{i=1}^n \frac{B^\top(x_i)JA(x_i)tt^\top A^\top(x_i)J^\top u_i}{\|JA(x_i)t\|^2}. \quad (\text{A.7})$$

As in Chiuso et al. 2000, we drop the denominator terms. This gives us: The first term we need to consider is the 3 by 3 matrix we need to invert.

$$\hat{\omega}(t) = G(t)^{-1}H(t) \quad (\text{A.8})$$

$$G(t) = \sum_{i=1}^n B^\top(x_i)JA(x_i)tt^\top A^\top(x_i)J^\top B(x_i) \quad (\text{A.9})$$

$$H(t) = \sum_{i=1}^n B^\top(x_i)JA(x_i)tt^\top A^\top(x_i)J^\top u_i. \quad (\text{A.10})$$

We focus on $G(t)$ first. We can write this out in terms of quadratic terms of tt^\top by introducing the matrices S^{ij} , defined as:

$$S_{kl}^{ij} = \begin{cases} 1 & \text{if } i = k, j = l, \text{ or } i = l, j = k \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}
G(t) &= t_1^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{11} A^\top(x_i) J^\top B(x_i) \right) \\
&\quad + t_1 t_2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{12} A^\top(x_i) J^\top B(x_i) \right) \\
&\quad + t_1 t_3 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{13} A^\top(x_i) J^\top B(x_i) \right) \\
&\quad + t_2^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{22} A^\top(x_i) J^\top B(x_i) \right) \\
&\quad + t_2 t_3 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{23} A^\top(x_i) J^\top B(x_i) \right) \\
&\quad + t_3^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{33} A^\top(x_i) J^\top B(x_i) \right) \\
&= \sum_{i < j} t_i t_j G^{ij},
\end{aligned}$$

where G^{ij} is defined appropriately. We will use G_k^{ij} to denote the k^{th} column of G^{ij} . We know that the inverse of a 3 by 3 matrix with columns c_1, c_2, c_3 has an inverse given by

$$\frac{1}{\det([c_1 \ c_2 \ c_3])} \begin{bmatrix} (c_2 \times c_3)^\top \\ (c_3 \times c_1)^\top \\ (c_1 \times c_2)^\top \end{bmatrix}.$$

We also know that the cross product is bi-linear, so from this we can write out the inverse of G

analytically:

$$\begin{aligned}
G^{-1}(t) &= \frac{1}{\det(G(t))} \begin{bmatrix} \left(\left(\sum_{i<j} t_i t_j G_2^{ij} \right) \times \left(\sum_{k<l} t_k t_l G_3^{kl} \right) \right)^\top \\ \left(\left(\sum_{i<j} t_i t_j G_3^{ij} \right) \times \left(\sum_{k<l} t_k t_l G_1^{kl} \right) \right)^\top \\ \left(\left(\sum_{i<j} t_i t_j G_1^{ij} \right) \times \left(\sum_{k<l} t_k t_l G_2^{kl} \right) \right)^\top \end{bmatrix} \\
&= \frac{1}{\det(G(t))} \begin{bmatrix} \sum_{i<j,k<l} t_i t_j t_k t_l \left(G_2^{ij} \times G_3^{kl} \right)^\top \\ \sum_{i<j,k<l} t_i t_j t_k t_l \left(G_3^{ij} \times G_1^{kl} \right)^\top \\ \sum_{i<j,k<l} t_i t_j t_k t_l \left(G_1^{ij} \times G_2^{kl} \right)^\top \end{bmatrix} \\
&= \frac{1}{\det(G(t))} \sum_{i<j,k<l} t_i t_j t_k t_l \begin{bmatrix} \left(G_2^{ij} \times G_3^{kl} \right)^\top \\ \left(G_3^{ij} \times G_1^{kl} \right)^\top \\ \left(G_1^{ij} \times G_2^{kl} \right)^\top \end{bmatrix}.
\end{aligned}$$

The terms in the matrix component become a 4^{th} degree polynomial of 3 variables with 15 terms (after grouping) with matrix coefficients. We can also compute the determinant explicitly using the fact that the determinant of a 3×3 matrix is the triple product of its columns.

$$\begin{aligned}
\det(G(t)) &= (G_2(t) \times G_3(t))^\top G_1(t) \\
&= \left(\sum_{i<j,k<l} t_i t_j t_k t_l \left(G_2^{ij} \times G_3^{kl} \right) \right)^\top \left(\sum_{p<q} t_p t_q G_1^{pq} \right) \\
&= \sum_{i<j,k<l,p<q} t_i t_j t_k t_l t_p t_q \left(\left(G_2^{ij} \times G_3^{kl} \right)^\top G_1^{pq} \right)
\end{aligned}$$

After grouping terms, this becomes a 6^{th} degree polynomial with 28 terms. This makes each element of G^{-1} a 6^{th} degree rational function.

In a similar fashion, we find the expression

$$\begin{aligned}
H(t) &= t_1^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{11} A^\top(x_i) J^\top u_i \right) \\
&\quad + t_1 t_2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{12} A^\top(x_i) J^\top u_i \right) \\
&\quad + t_1 t_3 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{13} A^\top(x_i) J^\top u_i \right) \\
&\quad + t_2^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{22} A^\top(x_i) J^\top u_i \right) \\
&\quad + t_2 t_3 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{23} A^\top(x_i) J^\top u_i \right) \\
&\quad + t_3^2 \left(\sum_{i=1}^n B^\top(x_i) J A(x_i) S^{33} A^\top(x_i) J^\top u_i \right) \\
&= \sum_{i < j} t_i t_j H^{ij}.
\end{aligned}$$

This gives the final form of the equation:

$$\hat{\omega}(t) = \frac{1}{\det(G(t))} \sum_{i < j, k < l, p < q} t_i t_j t_k t_l t_p t_q \begin{bmatrix} \left(G_2^{ij} \times G_3^{kl} \right)^\top \\ \left(G_3^{ij} \times G_1^{kl} \right)^\top \\ \left(G_1^{ij} \times G_2^{kl} \right)^\top \end{bmatrix} H^{pq}. \quad (\text{A.11})$$

We are left with $\hat{\omega}(t)$ as a 6^{th} degree rational function of V , meaning it has 28 terms in the numerator and denominator for each element.

A.4.2. Expression of cost function

First, we express the cost function as:

$$f(t) = \sum_i \left(t^\top A^\top(x_i) J^\top (B(x_i) \hat{\omega}(t) - u_i) \right)^2 \quad (\text{A.12})$$

Now we expand and simplify this by plugging in the definitions given above for G and H :

$$\begin{aligned}
f(t) &= \sum_i \left(t^\top A^\top(x_i) J^\top (B(x_i) \hat{\omega}(t) - u_i) \right)^2 \\
&= \sum_i (B(x_i) \hat{\omega}(t) - u_i)^\top J A(x_i) t t^\top A^\top(x_i) J^\top (B(x_i) \hat{\omega}(t) - u_i) \\
&= \hat{\omega}(t)^\top \left(\sum_i B^\top(x_i) J A(x_i) t t^\top A^\top(x_i) J^\top B(x_i) \right) \hat{\omega}(t) \\
&\quad - \left(\sum_i B^\top(x_i) J A(x_i) t t^\top A^\top(x_i) J^\top u_i \right)^\top \hat{\omega}(t) \\
&\quad + t^\top \left(\sum_i A^\top(x_i) J^\top u_i u_i^\top J A(x_i) \right) t \\
&= \hat{\omega}(t)^\top G(t) \hat{\omega}(t) - 2H(t)^\top \hat{\omega}(t) + t^\top S t \\
&= \hat{\omega}(t)^\top G(t) (G^{-1}(t) H(t)) - 2H(t)^\top \hat{\omega}(t) + t^\top S t \\
&= \hat{\omega}(t)^\top H(t) - 2H(t)^\top \hat{\omega}(t) + t^\top S t \\
&= t^\top S t - H(t)^\top \hat{\omega}(t),
\end{aligned}$$

where

$$S = \sum_i A^\top(x_i) J^\top u_i u_i^\top J A(x_i).$$

This gives us the final equation

$$f(t) = t^\top S t - H(t)^\top \hat{\omega}(t). \tag{A.13}$$

APPENDIX B : Supplemental material: Understanding image motion with group representations

B.1. Additional Experiments

Here, we expand on the comparison to the self-supervised optical flow baseline given in Table 2. Our method performs equivalently to the Flow+PCA method using the top four to five principal components (which account for most of the motion variance on KITTI, as shown in Figure 23). The marginal improvement in Flow+PCA appears to sharply drop off beginning around four to five principal components as well. As we saw before, the most dramatic increases in performance come from the Z component of translation and the Y component of rotation, which are the axes of the dominant motion and where chance error is highest.

We note that egomotion estimation benefits greatly from maintaining information about spatial position. Methods using flow fields maintain the information by explicitly representing local motion at each position of the image, but our method is global and does not. KITTI visual odometry is characterized by stereotyped depth and is reasonably modeled as rigid. Under these circumstances, camera translation and rotation can be estimated from a full flow field nearly linearly (Heeger and Jepson 1992). Consistent with this explanation, flow principal components appear to capture both the dominant motions exhibited by the vehicle on this dataset and the stereotyped depth configuration of KITTI (Figure 24). The good performance of Flow+PCA here highlights the clear advantage of domain-restricted models and learning rules in a setting where those domain restrictions are appropriate. Our learning rule and model do not make these more restrictive assumptions but still performs reasonably in this setting.

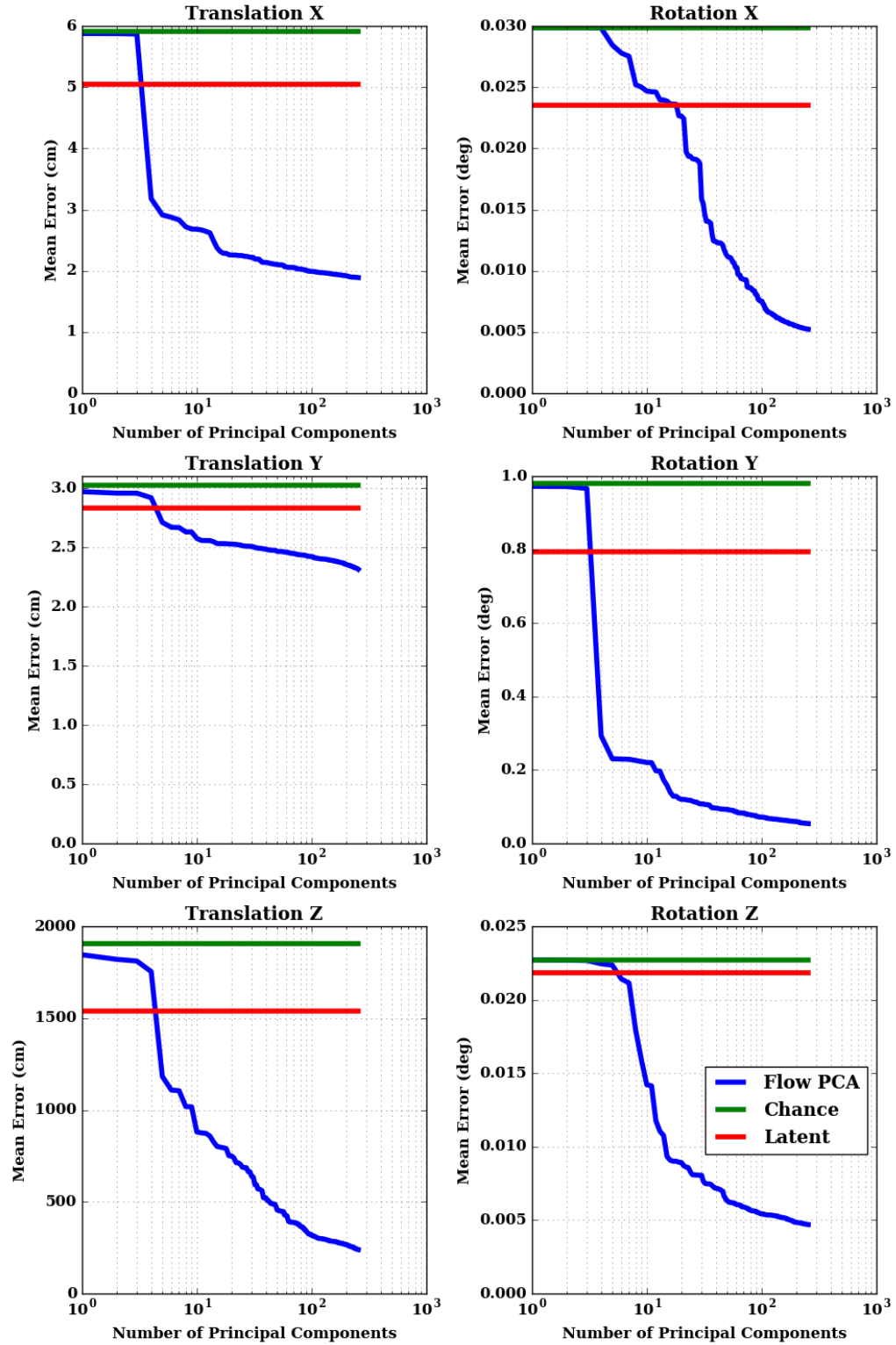


Figure 22: Error on egomotion regression from self-supervised flow PCA as a function of the number of principal components included. Horizontal lines reflect our method (latent, shown in red) and a chance baseline (shown in green).

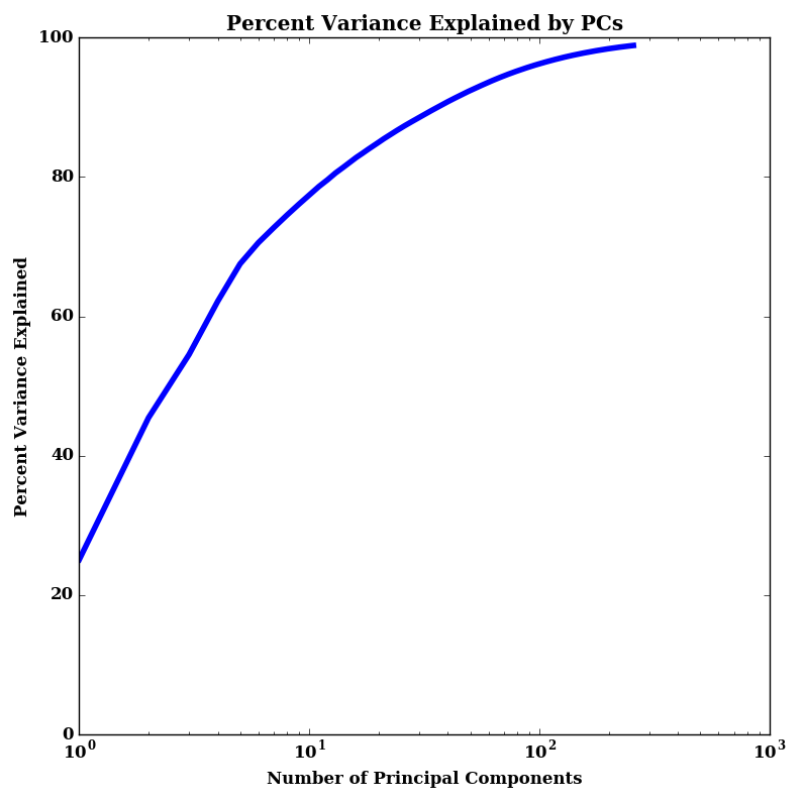


Figure 23: Cumulative percent variance explained of the optical flow in KITTI odometry as a function of the number of principal components included. 67% of the variance is explained by the first 5 components; 90% of the variance is explained by the first 40 principal components.

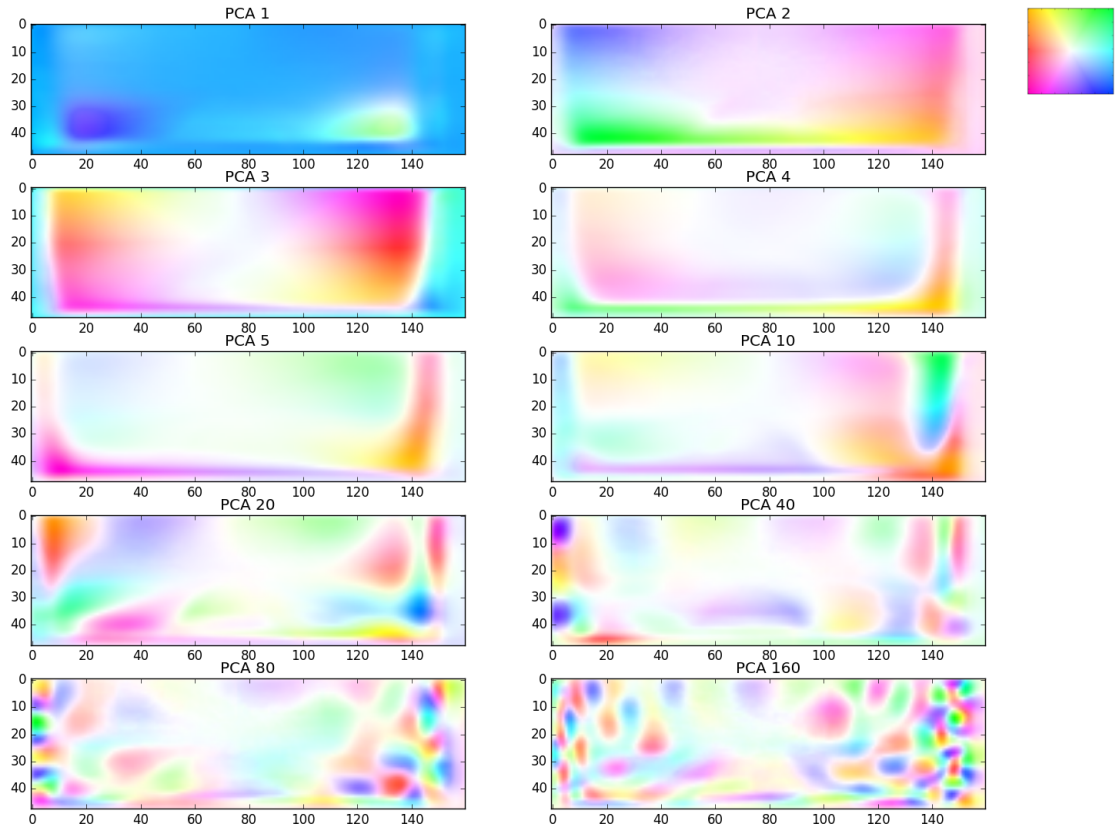


Figure 24: Representative principal components of optical flow on the KITTI odometry dataset. The first few components capture the dominant motions (forward and left/right turning) and reflect the stereotypical depth structure of KITTI.

APPENDIX C : Supplemental material: Predicting the future with transformational states

C.1. Video results

In videos included on the project website (https://daniilidis-group.github.io/transformational_states), we visualize prediction results from our model on a large number of sequences from Moving MNIST, KTH, and UCF101. Videos are chosen randomly from the three datasets. In all cases, we show the full sequence given as input to the network (10 frames for Moving MNIST and KTH, 2 frames for UCF101) and the ground truth future sequence along with the network prediction (10 frames for Moving MNIST and KTH, 1 frame for UCF101). Videos are looped and the frames predicted by the network are highlighted in green for clarity.

C.2. Network architectures

Full architectures for CNN encoders, CNN decoders, and the two components of the full RNN core (CNN_Φ and RNN) are given for the three datasets below. Parameters for convolutional (Conv), transposed convolutional (TransposedConv), and convolutional LSTM (ConvLSTM) layers are specified as {input feature map spatial dimensions, filter size, number of filters, convolution stride}. Other network elements either have no parameters (tanh and sigmoid activation functions) or always use the default Tensorflow parameter settings (batch norm (BN), leaky rectified linear unit (LReLU)).

Network components are wired together as in Figure 14. Weighted residual connections are used identically in the KTH and UCF architectures. The Moving MNIST architecture omits the residual connection to the output image, but is otherwise identical. Note that CNN encoders and CNN_Φ components include an output tanh nonlinearity to make it easier for the network to match their output distributions to that of a ConvLSTM.

Moving MNIST:

Encoder

$\text{Conv}\{64 \times 64, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{32 \times 32, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{16 \times 16, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{8 \times 8, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow \text{BN} \rightarrow \tanh$

Decoder

$\text{TransposedConv}\{4 \times 4, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{8 \times 8, 4 \times 4, 96, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{16 \times 16, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{32 \times 32, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{64 \times 64, 4 \times 4, 1, 1\} \rightarrow \text{sigmoid}$

RNN

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 64, 1\} \rightarrow$

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 64, 1\} \rightarrow$

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 64, 1\}$

CNN_Φ

$\text{Conv}\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{4 \times 4, 4 \times 4, 64, 1\} \rightarrow \tanh$

KTH:

Encoder

$\text{Conv}\{128 \times 128, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{64 \times 64, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{16 \times 16, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{4 \times 4, 4 \times 4, 256, 1\} \rightarrow \text{BN} \rightarrow \tanh$

Decoder

$\text{TransposedConv}\{4 \times 4, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{16 \times 16, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{32 \times 32, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{64 \times 64, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{128 \times 128, 4 \times 4, 1, 1\} \rightarrow \tanh$

RNN

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 128, 1\} \rightarrow$

ConvLSTM $\{4 \times 4, 3 \times 3, 128, 1\} \rightarrow$

ConvLSTM $\{4 \times 4, 3 \times 3, 128, 1\}$

CNN $_{\Phi}$

Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow$ LReLU \rightarrow

Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow$ LReLU \rightarrow

Conv $\{4 \times 4, 4 \times 4, 128, 1\} \rightarrow$ tanh

UCF:

Encoder

$\text{Conv}\{256 \times 256, 4 \times 4, 64, 2\} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{128 \times 128, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{64 \times 64, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{16 \times 16, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{Conv}\{4 \times 4, 4 \times 4, 512, 1\} \rightarrow \text{BN} \rightarrow \tanh$

Decoder

$\text{TransposedConv}\{4 \times 4, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{8 \times 8, 4 \times 4, 512, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{16 \times 16, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{32 \times 32, 4 \times 4, 256, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{64 \times 64, 4 \times 4, 128, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{128 \times 128, 4 \times 4, 64, 2\} \rightarrow \text{BN} \rightarrow \text{LReLU} \rightarrow$

$\text{TransposedConv}\{256 \times 256, 4 \times 4, 1, 2\} \rightarrow \tanh$

RNN

$\text{ConvLSTM}\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$

ConvLSTM $\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$

ConvLSTM $\{4 \times 4, 3 \times 3, 256, 1\}$

CNN $_{\Phi}$

Conv $\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$ LReLU \rightarrow

Conv $\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$ LReLU \rightarrow

Conv $\{4 \times 4, 3 \times 3, 256, 1\} \rightarrow$ tanh

C.3. Comparison to other prediction models

In Table 7, we compare details of the architecture and training configurations used in various recently proposed architectures for future prediction. Most notably, we produce future predictions without re-encoding predicted images as input for the encoder network, without directly copying from the input sequence, and without using GANs at any point in network training. The gradient difference loss (GDL) is defined in Mathieu et al. 2016.

We strongly encourage the reader to investigate the cited papers for more details: this table is intended only as a road map to the very interesting and growing literature on future prediction.

C.4. Ablation studies

Here, we present qualitative results from ablations of the proposed architecture on Moving MNIST (Figure 25) and KTH (Figures 26 and 27). On Moving MNIST, we show the results of training the model with and without weighted residual connections. Moving MNIST images are fairly simple, so residual connections do not lead to as large an improvement in performance as on datasets with real-world image statistics.

On KTH, we compare the full model against models trained (i) with a ConvLSTM core instead of the full RNN core described in the main paper, (ii) using residual connections directly from the last input time step instead of the previous time step (i.e. the decoder at time $t = T + k$ receives skip

Table 7: Comparison of sequence prediction model components and training configurations.

	Uses skip connections or copies past?	Re-encodes images to generate predictions after $t=T+1$?	Uses LSTMs?	Uses additional labels or training?	Uses GANs?	Loss
BeyondMSE (Mathieu et al. 2016)	Uses multi-scale Laplacian pyramid on full sequence (re-)encoding	Yes	No	No	GAN on predicted images	MSE, GDL, GAN
MCNet (Villegas et al. 2017a)	Skips from previous frame and difference image re-encoding	Re-encodes images and re-computes difference images	ConvLSTM on difference image encoding	No	GAN on predicted images	MSE, GDL, GAN
DRNet (Denton and Birodkar 2017)	Copies content vector from last time step	No	LSTM on input sequence embedding	Encoder output trained to disentangle content from pose, content to remain static over a sequence	GAN to disentangle content and pose	Two-stage training: (1) GAN, (2) MSE
SVG-LP (Denton and Fergus 2018)	Skips from last input frame encoding	Yes	LSTM on encoder output and LSTM on learned prior	No	No	MSE, KL divergence between model and learned prior
SNA (Ebert et al. 2017)	Skips from previous frame re-encoding and from last input frame	Yes	ConvLSTM layers throughout encoder and decoder	Uses control state and action as additional input	No	MSE
Dual Motion GAN (Liang et al. 2017)	No	Yes	ConvLSTM on encoder output	Trains network to predict current and future optical flow	GAN on predicted images and predicted current and future flow	GAN, VAE KL divergence
DVF (Liu et al. 2017)	No	Yes	No	No	No	L1, total variation losses
PredNet (Lotter et al. 2017)	No	Yes	ConvLSTMs throughout architectures	No	No	Predictive coding L1 loss
Adversarial Transformer (Vondrick and Torralba 2017)	Predicted images given as interpolation of last input image pixels	No, last input image directly transformed	No	No	Uses conditional GAN on predicted sequences	GAN
Ours	Masked residual from previous decoder state	No	ConvLSTM on encoder output	No	No	MSE

connections from the encoder at time $t = T$ instead of the decoder at time $t = T + k - 1$), (iii) with no skip or residual connections. The full model best captures image motion while also leading to better background in-painting.

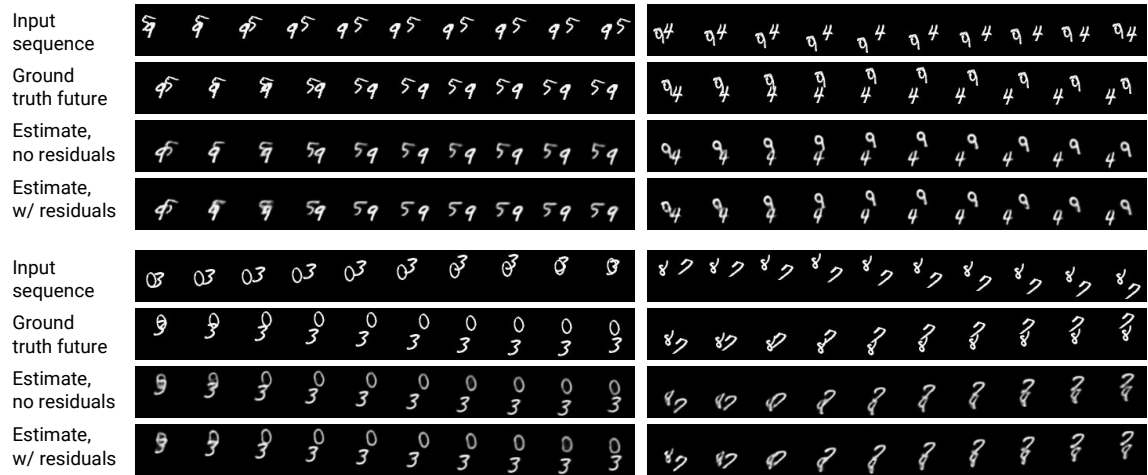


Figure 25: Comparison of Moving MNIST results on architectures with and without residual connections. The model labeled “no residuals” has no skip or residual connections of any kind. The model labeled “w/ residuals” is the full model described in the paper. The model without weighted residual connections produces good predictions, but including these connections produces crisper results, especially at early prediction time steps. Both architectures reliably capture digit identity, even after the digits overlap.

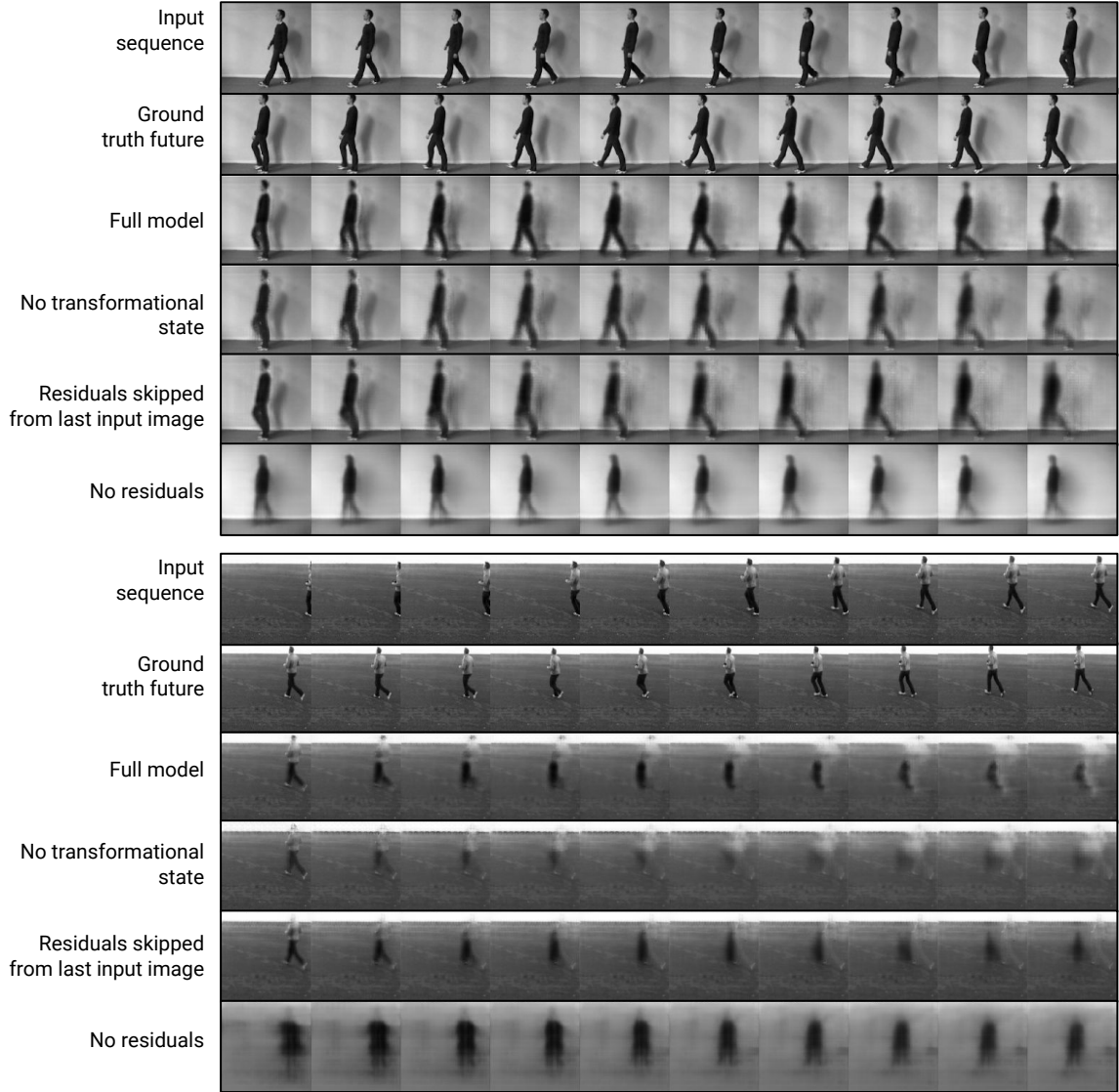


Figure 26: Comparison of KTH results on models with architectural ablations. (i) “No transformational state”: the RNN core omits the CNN_Φ and includes only ConvLSTM components. (ii) “Residuals skipped from last input image”: each decoder directly receives residual input from the encoder at the last input time step ($t = T$) instead of the previous decoder time step. Weighted residuals are still used. (iii) “No residuals”: no residual or skip connections of any kind are used. The second sequence shown here is very challenging for all models. The full model produces better motion (notice the motion of the legs) and less prominent ghosting artifacts than ablations.

All models were trained with the hyperparameters used to train the model described in the paper. On KTH, this produced good results for all models including residual connections. We have seen

qualitatively better motion on the model without residuals using different hyperparameters, but the results shown here are representative of the difference between models. Including residual connections led to dramatically better results on background prediction, but the model without residual connections appears to model motion reasonably well in some cases.

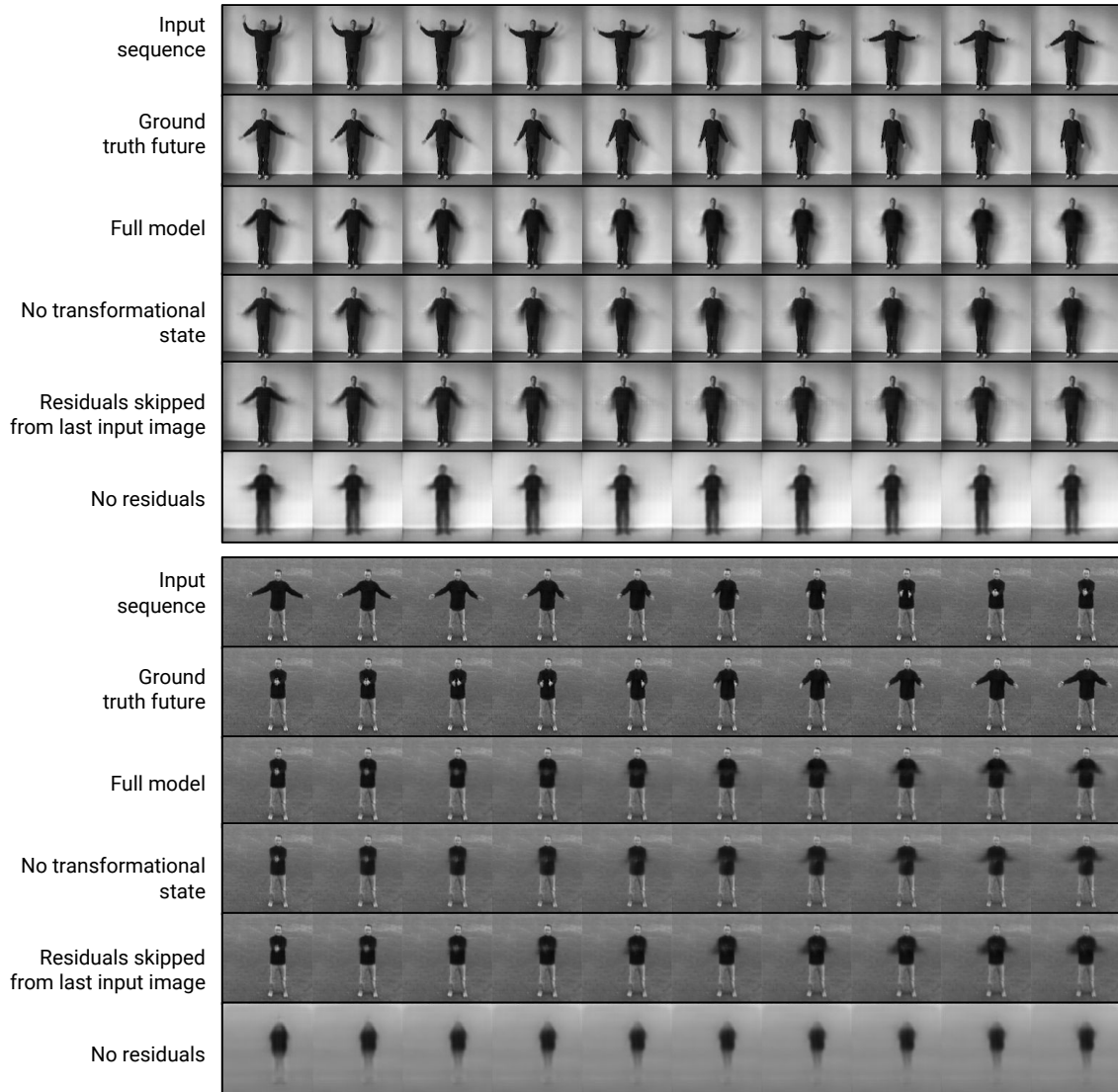


Figure 27: Additional comparisons of KTH results on models with architectural ablations. See Figure 26 caption for explanation of ablations.

BIBLIOGRAPHY

- A. Achille and S. Soatto. Emergence of invariance and disentangling in deep representations. In *arXiv e-prints*, 2017.
- E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(2):284–299, 1985.
- P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.
- P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989.
- D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: Shape completion and animation of people. *ACM Transactions on Graphics*, 24(3):408–416, 2005.
- F. Anselmi, J. Z. Leibo, L. Rosasco, J. Mutch, A. Tacchetti, and T. Poggio. Unsupervised learning of invariant representations. *Theoretical Computer Science*, 633:112 – 121, 2016.
- G. Berkeley. *An Essay Towards a New Theory of Vision*. 1709.
- M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *ICCV*, 1993.
- M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996.
- A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, 1987.
- A. Borst, J. Haag, and D. F. Reiff. Fly motion vision. *Annual Review of Neuroscience*, 33(1):49–70, 2010.
- B. D. Brabandere, X. Jia, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NIPS*, 2016.
- C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *CVPR*, 2000.
- P.-J. Bristeau, F. Callou, D. Vissire, and N. Petit. The navigation and control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes*, 44(1):1477 – 1484, 2011.
- N. A. Browning. A neural circuit for robust time-to-contact estimation based on primate MST. *Neural Computation*, 24(11):2946–2963, 2012.
- T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

- A. Bubic, D. Y. Von Cramon, and R. Schubotz. Prediction, cognition and the brain. *Frontiers in Human Neuroscience*, 4, 2010.
- J. Burge and W. S. Geisler. Optimal speed estimation in natural image movies predicts human performance. *Nature Communications*, 6:7900, 2015.
- J. Bütepage, M. J. Black, D. Kragic, and H. Kjellström. Deep representation learning for human motion prediction and classification. In *CVPR*, 2017.
- A. Byravan, F. Leeb, F. Meier, and D. Fox. SE3-Pose-Nets: Structured deep dynamics models for visuomotor planning and control. In *arXiv e-prints*, 2017.
- S. A. Cadena, G. H. Denfield, E. Y. Walker, L. A. Gatys, A. S. Tolia, M. Bethge, and A. S. Ecker. Deep convolutional models improve predictions of macaque V1 responses to natural images. *bioRxiv*, 2017.
- C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLOS Computational Biology*, 10(12):1–18, 2014.
- M. Carandini and D. J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13:51–62, 2011.
- X. Chen, G. C. DeAngelis, and D. E. Angelaki. Flexible egocentric and allocentric representations of heading signals in parietal cortex. *Proceedings of the National Academy of Sciences*, 115(14): E3305–E3312, 2018.
- M. Chessa, N. V. K. Medathati, G. Masson, F. Solari, and P. Kornprobst. Decoding MT motion response for optical flow estimation: An experimental evaluation. In *EUSIPCO*, 2015.
- S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed. Recurrent Environment Simulators. In *ICLR*, 2017.
- A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion: Local ambiguities and global estimates. *International Journal of Computer Vision*, 39(3):195–228, 2000.
- S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.
- P. S. Churchland, V. Ramachandran, and T. S. Sejnowski. A critique of pure vision. In C. Koch and J. L. Davis, editors, *Computational neuroscience. Large-scale neuronal theories of the brain*. MIT Press, 1994.
- R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A MATLAB-like environment for machine learning. In *NIPS Workshops*, 2011.
- P. I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Springer, 2011.

- G. Costante and T. A. Ciarfuglia. LS-VO: Learning Dense Optical Subspace for Robust Visual Odometry Estimation. In *arXiv e-prints*, 2017.
- G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia. Exploring representation learning with cnns for frame-to-frame ego-motion estimation. In *ICRA*, 2016.
- F. Cricri, X. Ni, M. Honkala, E. Aksu, and M. Gabbouj. Video ladder networks. In *arXiv e-prints*, 2016.
- K. Daniilidis and H.-H. Nagel. Analytical results on error sensitivity of motion estimation from two views. *Image and Vision Computing*, 8(4):297–303, 1990.
- E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *arXiv e-prints*, 2018.
- E. L. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017.
- J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.
- C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.
- A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. In *ICLR*, 2017.
- R. Dubner and S. Zeki. Response properties and receptive fields of cells in an anatomically defined region of the superior temporal sulcus in the monkey. *Brain Research*, 35(2):528 – 532, 1971.
- C. Duffy and R. Wurtz. Response of monkey MST neurons to optic flow stimuli with shifted centers of motion. *Journal of Neuroscience*, 15(7):5192–5208, 1995.
- F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. In *CoRL*, 2017.
- G. Farneäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian Conference on Image Analysis*, 2003.
- C. Finn, I. J. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.
- P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning optical flow with convolutional networks. In *CVPR*, 2015.
- C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *ICRA*, 2014.
- A. Fragkiadaki, B. Seybold, R. Sukthankar, S. Vijayanarasimhan, and S. Ricco. Self-supervised learning of structure and motion from video. In *arXiv e-prints*, 2017.

- K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015.
- F. Fraundorfer and D. Scaramuzza. Visual odometry. Part II: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.
- J. Fredriksson, O. Enqvist, and F. Kahl. Fast and reliable two-view translation estimation. In *CVPR*, 2014.
- J. Fredriksson, V. Larsson, and C. Olsson. Practical robust two-view translation estimation. In *CVPR*, 2015.
- J. Freeman, C. M. Ziemba, D. J. Heeger, E. P. Simoncelli, and J. A. Movshon. A functional and perceptual signature of the second visual area in primates. *Nature Neuroscience*, 16:974–981, 2013.
- K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- R. Garg, G. Carneiro, and I. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.
- A. Geiger, J. Ziegler, and C. Stiller. StereoScan: Dense 3D reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, 2011.
- A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, 2012.
- D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(3):367–383, 1992.
- G. Gergely, H. Bekkering, and I. Király. Rational imitation in preverbal infants. *Nature*, 415:755, 2002.
- J. J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.
- M. A. Giese and T. Poggio. Neural mechanisms for the recognition of biological movements. *Nature Reviews Neuroscience*, 4:179 – 192, 2003.
- M. S. Gizzi, E. Katz, R. A. Schumer, and J. A. Movshon. Selectivity for orientation and direction of motion of single neurons in cat striate and extrastriate visual cortex. *Journal of Neurophysiology*, 63(6):1529–1543, 1990.
- J. I. Gold and M. N. Shadlen. Representation of a perceptual decision in developing oculomotor commands. *Nature*, 2000.
- M. A. Goodale and A. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20 – 25, 1992.

- I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. In *arXiv e-prints*, 2017.
- I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
- I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- R. Goroshin, M. Mathieu, and Y. LeCun. Learning to linearize under uncertainty. In *NIPS*, 2015.
- M. Graziano, R. Andersen, and R. Snowden. Tuning of MST neurons to spiral motions. *Journal of Neuroscience*, 14(1):54–67, 1994.
- C. Grefkes and G. R. Fink. The functional organization of the intraparietal sulcus in humans and monkeys. *Journal of Anatomy*, 207(1):3–17, 2005.
- S. Grossberg, E. Mingolla, and C. C. Pack. A neural model of motion processing and visual navigation by cortical area MST. *Cerebral Cortex*, 9(8):878–95, 1999.
- Y. Gu, C. R. Fetsch, B. Adeyemo, G. C. DeAngelis, and D. E. Angelaki. Decoding of MSTd population activity accounts for variations in the precision of heading perception. *Neuron*, 66(4): 596 – 609, 2010.
- D. Ha and J. Schmidhuber. World Models. In *arXiv e-prints*, 2018.
- S. B. Hamed, W. Page, C. Duffy, and A. Pouget. MSTd neuronal basis functions for the population encoding of heading direction. *Journal of Neurophysiology*, 90(2):549–558, 2003.
- B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.
- R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- B. Hassenstein and W. E. Reichardt. Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungspersonzeption des Rüsselkäfers *Chlorophanus*. *Zeitschrift für Naturforschung B*, 11b:513–524, 1956.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- D. J. Heeger and A. D. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.
- E. C. Hildreth. Recovering heading for visually-guided navigation. *Vision Research*, 32(6):1177–1192, 1992.
- E. C. Hildreth and C. Koch. The analysis of visual motion: From computational theory to neuronal mechanisms. *Annual Review of Neuroscience*, 10(1):477–533, 1987.

- G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. In *arXiv e-prints*, 2012.
- J. Ho and S. Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977.
- B. K. P. Horn. *Robot Vision*. MIT Press, 1986.
- B. K. P. Horn. Motion fields are hardly ever ambiguous. *International Journal of Computer Vision*, 1(3):259–274, 1988.
- B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981.
- G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.
- P. J. Huber. *Robust statistics*. Springer, 2011.
- S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
- M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*, 2017.
- A. Jaegle, S. Phillips, and K. Daniilidis. Fast, robust, continuous monocular egomotion computation. In *ICRA*, 2016.
- A. Jaegle, S. Phillips, D. Ippolito, and K. Daniilidis. Understanding image motion with group representations. In *ICLR*, 2018a.
- A. Jaegle, O. Rybkin, K. G. Derpanis, and K. Daniilidis. Predicting the Future with Transformational States. In *arXiv e-prints*, 2018b.
- D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. *ICCV*, 2015.
- A. D. Jepson and D. J. Heeger. Subspace methods for recovering rigid motion II: Theory. Technical Report RBCV-TR-90-36, University of Toronto, 1990.
- A. D. Jepson and D. J. Heeger. A fast subspace algorithm for recovering rigid motion. In *IEEE Workshop on Visual Motion*, 1991.
- H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.

- J. P. Jones and L. A. Palmer. The two-dimensional spatial structure of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1187–1211, 1987.
- N. Kalchbrenner, A. van den Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. In *ICML*, 2017.
- K. Kanatani. 3-D interpretation of optical flow by renormalization. *International Journal of Computer Vision*, 11(3):267–282, 1993.
- A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.
- C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *ICRA*, 2013.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.
- K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.
- E. Kobatake and K. Tanaka. Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex. *Journal of Neurophysiology*, 71(3):856–867, 1994.
- K. R. Konda and R. Memisevic. A unified approach to learning depth and motion features. *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)*, 2014.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.
- V. A. Lamme, H. Supr, and H. Spekreijse. Feedforward, horizontal, and feedback processing in the visual cortex. *Current Opinion in Neurobiology*, 8(4):529 – 535, 1998.
- I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.
- Q. V. Le. Building high-level features using large scale unsupervised learning. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.
- D. Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.
- Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521, 2015.
- M. Leinweber, D. R. Ward, J. M. Sobczak, A. Attinger, and G. B. Keller. A sensorimotor circuit in mouse cortex for visual flow predictions. *Neuron*, 95(6):1420–1432, 2017.
- B. Li, B. Li, Y. Chen, L. Wang, and Y. Diao. Response properties of PMLS and PLLS neurons to simulated optic flow patterns. *European Journal of Neuroscience*, 12(5):1534–1544, 2000.
- T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics*, 36(6):194:1–194:17, 2017.

- X. Liang, L. Lee, W. Dai, and E. P. Xing. Dual motion GAN for future-flow embedded video prediction. In *ICCV*, 2017.
- M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *NIPS*, 2002.
- Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017.
- J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 208(1173):385–397, 1980.
- M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics*, 34(6):248:1–248:16, 2015.
- W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017.
- D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- C. Lu, M. Hirsch, and B. Schölkopf. Flexible spatio-temporal networks for video prediction. In *CVPR*, 2017.
- P. Luc, N. Neverova, C. Couprie, J. Verbeek, and Y. LeCun. Predicting deeper into the future of semantic segmentation. In *ICCV*, 2017.
- P. Luc, C. Couprie, Y. LeCun, and J. Verbeek. Predicting future instance segmentations by forecasting convolutional features. In *arXiv e-prints*, 2018.
- B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.
- Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry. *An invitation to 3-D vision: from images to geometric models*. Springer, 2004.
- J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017.
- M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.
- J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. In *arXiv e-prints*, 2017.

- P. J. Mineault, F. A. Khawaja, D. A. Butts, and C. C. Pack. Hierarchical processing of complex motion along the primate dorsal visual pathway. *Proceedings of the National Academy of Sciences*, 109(16):E972–E980, 2012.
- I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *ECCV*, 2016.
- A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick. On the importance of single directions for generalization. In *ICLR*, 2018.
- J. A. Movshon and W. T. Newsome. Visual response properties of striate cortical neurons projecting to area MT in macaque monkeys. *Journal of Neuroscience*, 16(23):7733–7741, 1996.
- R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011.
- A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.
- W. T. Newsome, K. H. Britten, and J. A. Movshon. Neuronal correlates of a perceptual decision. *Nature*, 341:52 – 54, 1989.
- M. H. Nguyen and F. D. la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.
- D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.
- J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.
- C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 2018. <https://distill.pub/2018/building-blocks>.
- J. Oliensis. The least-squares error for structure from infinitesimal motion. *International Journal of Computer Vision*, 61(3):259–299, 2005.
- B. Ouellette, K. Minville, J. Faubert, and C. Casanova. Simple and complex visual motion response properties in the anterior medial bank of the lateral suprasylvian cortex. *Neuroscience*, 123(1):231 – 245, 2004.
- C. C. Pack, S. Grossberg, and E. Mingolla. A neural model of smooth pursuit control and motion perception by cortical area MST. *Journal of Cognitive Neuroscience*, 13(1):102–120, 2001.
- C. C. Pack, M. S. Livingstone, K. R. Duffy, and R. T. Born. End-stopping and the aperture problem. *Neuron*, 39(4):671–680, 2003.

- L. A. Palmer, A. C. Rosenquist, and R. J. Tusa. The retinotopic organization of lateral suprasylvian visual areas in the cat. *Journal of Comparative Neurology*, 177(2):237–256, 1978.
- S. E. Palmer, O. Marre, M. J. Berry, and W. Bialek. Predictive information in a sensory population. *Proceedings of the National Academy of Sciences*, 112(22):6908–6913, 2015.
- D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *CVPR*, 2017.
- V. Pătrăucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *ICLR Workshop*, 2016.
- G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3D human pose. In *CVPR*, 2017.
- J. A. Perrone and L. S. Stone. A model of self-motion estimation within primate extrastriate visual cortex. *Vision Research*, 34(21):2917 – 2938, 1994.
- J. A. Perrone and L. S. Stone. Emulating the visual receptive-field properties of MST neurons with a template model of heading estimation. *Journal of Neuroscience*, 18(15):5958–5975, 1998.
- G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics*, 34(4):120:1–120:14, 2015.
- C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.
- A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.
- A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017.
- M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. In *arXiv e-prints*, 2014.
- M. B. Reiser and M. H. Dickinson. Visual motion speed determines a behavioral switch from forward flight to expansion avoidance in *Drosophila*. *The Journal of Experimental Biology*, 216(4):719–732, 2013.
- Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.
- J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986.

- N. C. Rust and A. A. Stocker. Ambiguity and invariance: two fundamental challenges for visual processing. *Current Opinion in Neurobiology*, 20(3):382 – 388, 2010.
- N. C. Rust, O. Schwartz, J. A. Movshon, and E. P. Simoncelli. Spatiotemporal elements of macaque V1 receptive fields. *Neuron*, 46(6):945–956, 2005.
- N. C. Rust, V. Mante, E. P. Simoncelli, and J. A. Movshon. How MT cells analyze the motion of visual patterns. *Nature Neuroscience*, 9:1421–1431, 2006.
- T. M. Sanada and G. C. DeAngelis. Neural representation of motion-in-depth in area MT. *Journal of Neuroscience*, 34(47):15508–15521, 2014.
- R. Sasaki, D. E. Angelaki, and G. C. DeAngelis. Dissociation of self-motion and object motion by linear population decoding that approximates marginalization. *Journal of Neuroscience*, 37(46):11204–11219, 2017.
- A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox. On the information bottleneck theory of deep learning. In *ICLR*, 2018.
- J. W. Scannell, F. Sengpiel, M. J. Tovee, P. J. Benson, C. Blakemore, and M. P. Young. Visual motion processing in the anterior ectosylvian sulcus of the cat. *Journal of Neurophysiology*, 76(2):895–907, 1996.
- D. Scaramuzza and F. Fraundorfer. Visual odometry. Part I: The first 30 years and fundamentals. *IEEE Robotics & Automation Magazine*, 18(4):80–92, 2011.
- C. Schüldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In *ICPR*, 2004.
- N. Sedaghat, M. Zolfaghari, and T. Brox. Hybrid learning of optical flow and next frame prediction to boost optical flow in the wild. In *arXiv e-prints*, 2017.
- L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black. Optical Flow with Semantic Segmentation and Localized Layers. In *CVPR*, 2016.
- M. N. Shadlen and W. T. Newsome. Neural basis of a perceptual decision in the parietal cortex (area LIP) of the rhesus monkey. *Journal of Neurophysiology*, 86(4):1916–1936, 2001.
- X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.
- H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *J. Comput. Syst. Sci.*, 50(1):132–150, 1995.
- E. P. Simoncelli and D. J. Heeger. A model of neuronal responses in visual area MT. *Vision Research*, 38(5):743 – 761, 1998.

- K. Simonyan and A. Zissermann. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
- K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2013.
- J. M. Singer and D. L. Sheinberg. Temporal cortex neurons encode articulated actions as slow sequences of integrated poses. *Journal of Neuroscience*, 30(8):3133–3145, 2010.
- S. Soatto and R. Brockett. Optimal structure from motion: Local ambiguities and global estimates. In *CVPR*, 1998.
- F. Solari, M. Chessa, K. Medathati, and P. Kornprobst. What can we expect from a classical V1-MT feedforward architecture for optical flow estimation? Technical Report RR-8618, INRIA, 2014.
- S. Song, M. Chandraker, and C. Guest. Parallel, real-time monocular visual odometry. In *ICRA*, 2013.
- K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *arXiv e-prints*, 2012.
- E. Spelke, A. Phillips, and A. Woodward. Infants knowledge of object motion and human action. In D. Sperber, D. Premack, and A. J. Premack, editors, *Causal Cognition: A Multidisciplinary Debate*. Oxford University Press, 1996.
- N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015a.
- R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In *ICML Deep Learning Workshop*, 2015b.
- H. Stewenius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.
- H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.
- D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.
- A. Sunkara, G. C. DeAngelis, and D. E. Angelaki. Joint representation of translational and rotational components of optic flow in parietal cortex. *Proceedings of the National Academy of Sciences*, 113(18):5077–5082, 2016.
- I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

- A. Tacchetti, L. Isik, and T. Poggio. Invariant recognition drives neural representations of action sequences. *PLOS Computational Biology*, 13(12):1–20, 2017.
- A. Takemura, Y. Murata, K. Kawano, and F. A. Miles. Deficits in short-latency tracking eye movements after chemical lesions in monkey cortical areas MT and MST. *Journal of Neuroscience*, 27(3):529–541, 2007.
- K. Tanaka, K. Hikosaka, H. Saito, M. Yukie, Y. Fukada, and E. Iwai. Analysis of local and wide-field movements in the superior temporal visual areas of the macaque monkey. *Journal of Neuroscience*, 6(1):134–144, 1986.
- G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010.
- L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *ICLR*, 2016.
- N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, 2015.
- N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *The 37th annual Allerton Conference on Communication, Control, and Computing*, 1999.
- C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.
- C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.
- K. Toyama and T. Kozasa. Responses of Clare-Bishop neurones to three dimensional movement of a light stimulus. *Vision Research*, 22(5):571 – 574, 1982.
- D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. *ICCV*, 2015.
- D. Y. Tsao, W. A. Freiwald, R. B. H. Tootell, and M. S. Livingstone. A cortical region consisting entirely of face-selective cells. *Science*, 311(5761):670–674, 2006.
- S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017.
- S. Ullman. From simple innate biases to complex visual concepts. <https://cbmm.mit.edu/video/simple-innate-biases-complex-visual-concepts>, 2016.
- L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

- J. P. H. van Santen and G. Sperling. Elaborated reichardt detectors. *Journal of the Optical Society of America A*, 2(2):513–524, 1985.
- G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017a.
- R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, 2017b.
- C. Vondrick and A. Torralba. Generating the future with adversarial transformers. In *CVPR*, 2017.
- C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, 2016a.
- C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *CVPR*, 2016b.
- J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *ICCV*, 2015.
- J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017.
- X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.
- Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- E. Watanabe, A. Kitaoka, K. Sakamoto, M. Yasugi, and K. Tanaka. Illusory motion reproduced by deep neural networks trained for prediction. *Frontiers in Psychology*, 9, 2018.
- G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, M. Gemici, M. Reynolds, T. Harley, J. Abramson, S. Mohamed, D. Rezende, D. Saxton, A. Cain, C. Hillier, D. Silver, K. Kavukcuoglu, M. Botvinick, D. Hassabis, and T. Lillicrap. Unsupervised Predictive Memory in a Goal-Directed Agent. In *arXiv e-prints*, 2018.
- A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. Efficient dense scene flow from sparse or dense stereo data. In *ECCV*, 2008.
- L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

- J. Xiao, J.-X. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. In *ECCV*, 2004.
- D. L. K. Yamins and J. J. DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19:356–365, 2016.
- D. L. K. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.
- J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.
- J. J. Yu, W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV Workshops*, 2016.
- J. Yuen and A. Torralba. A data-driven approach for event prediction. In *ECCV*, 2010.
- C. Zach. Robust bundle adjustment revisited. In *ECCV*, 2014.
- C. Zach, A. Penate-Sanchez, and M.-T. Pham. A dynamic programming approach for fast and robust object pose recognition from range images. In *CVPR*, 2015.
- M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.
- R. S. Zemel and T. J. Sejnowski. A model for encoding multiple object motions and self-motion in area MST of primate visual cortex. *Journal of Neuroscience*, 18(1):531–547, 1998.
- K. Zeng, W. B. Shen, D. Huang, M. Sun, and J. C. Niebles. Visual forecasting by imitating dynamics in natural sequences. In *ICCV*, 2017.
- K. Zhang and T. J. Sejnowski. A theory of geometric constraints on neural activity for natural three-dimensional movement. *Journal of Neuroscience*, 19(8):3122–3145, 1999.
- T. Zhang and C. Tomasi. Fast, robust, and consistent camera motion estimation. In *CVPR*, 1999.
- T. Zhang and C. Tomasi. On the consistency of instantaneous rigid motion estimation. *International Journal of Computer Vision*, 46(1):51–79, 2002.
- B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015.
- T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.
- Y. Zhu, Z. Lan, S. D. Newsam, and A. G. Hauptmann. Guided optical flow learning. In *CVPR Workshops*, 2017.

- X. Zhuang, T. S. Huang, N. Ahuja, and R. M. Haralick. A simplified linear optic flow-motion algorithm. *Computer Vision, Graphics, and Image Processing*, 42(3):334–344, 1988.
- M. Zollhöffer, M. Nießner, S. Izadi, C. Rehmann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, and M. Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics*, 33(4):156, 2014.