Title
Learning Task-sufficient Representation of Video Dynamics

Author
Bei, Xinzhu

Publication Date
2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Learning Task-sufficient Representation of Video Dynamics

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Xinzhu Bei

2022

ABSTRACT OF THE DISSERTATION

Learning Task-sufficient Representation of Video Dynamics

by

Xinzhu Bei
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2022
Professor Stefano Soatto, Chair

This dissertation provides a generic solution to model dynamic systems whose hidden state and the transition model are unknown in practice. We build the *task-sufficient filtering* framework to maintain a finite, abstract, and learnable representation (memory) that is sufficient to update itself, casually and iteratively, and to predict downstream task variables of interest. We show our realization of the framework by recurrent neural networks as universally-approximating function classes to imitate the functionality of a state transition model and a task prediction model.

In addition, we provide practical methodologies to impose *generic priors* of the physical scene on the hidden representation. We leverage (lower-level) topological and regularity constraints of natural images, such as occlusion relations, to define object regions. Hence, we capture the motion priors associated with different (higher-level) semantic categories, that are combined to describe the dynamics of the whole scene.

The framework takes videos as sequential input streams and produces representations of video dynamics. We show the success of our framework by applying it to solve real-world computer vision tasks, including generic object tracking and video prediction. The

learned dynamic models are extensible to multiple circumstances requiring a dynamically and casually updated memory with uncertainty.

The dissertation of Xinzhu Bei is approved.

Yizhou Sun

Quanquan Gu

Yingnian Wu

Stefano Soatto, Committee Chair

University of California, Los Angeles

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Stefano Soatto, whose extraordinary achievements, unique insights into the field of computer vision, and rigorous attitude towards academic research have deeply influenced me. This provides a solid foundation for my 7-year doctoral research journey and future professional career.

I would like to thank my committee members, Professor Yingnian Wu, Professor Quanquan Gu, and Professor Yizhou Sun, for their valuable input throughout my Ph.D. studies. I would particularly like to thank Professor Quanquan Gu, as the tutor of two classes I TA'd, who guided me on being a good knowledge transmitter.

I would also like to thank all my labmates from UCLA Vision Lab. We encourage each other, learn from each other, and tackle complex academic problems together. Specifically, my collaborator Dr. Alessandro Achille, who contribute the theoretic proofs of task-sufficient dynamics in Chapter. 4; Dr. Yanchao Yang, who implemented the prediction module, and Dr. Brian Taylor, who implemented the object proposal module, of Chapter. 5; Dr. Yanchao Yang for co-authored Chapter. 6 as a version of [BYS21]. Also thank ARL W911NF-20-1-0158 and ONR N00014-17-1-2072 for the funding support of Chapter. 6.

In addition, I would like to thank my family that always offer me support and security coupled with unconditional love. Thank my parents for their life advice to help me make the right choices. Thank my grandparents, whom both departed during my Ph.D. study in the US, for their fostering in my childhood.

VITA

2011–2015 Received B.S. in Electronic and Information Engineering, Xi'an Jiaotong University, China. Program of Special Class for the Gifted Young of China.

2014 Research Intern, Cross-disciplinary Scholars in Science and Technology Program, UCLA.

2015–2022 Graduate Student Researcher, Computer Science Department, UCLA.

2018–2019 Teaching Assistant, Computer Science Department, UCLA. Taught CM146 (Introduction to Machine Learning) winter 2018, spring 2018; CS260 (Machine Learning), fall 2018; CS269 (Foundations of Deep Learning), winter 2019.

2019 Operations Research Scientist Intern, Facebook Inc.

2019–2022 Visiting Applied Research Scientist, Meta Platforms, Inc.

CHAPTER 1

Introduction

1.1 Learning cycle of human and machine

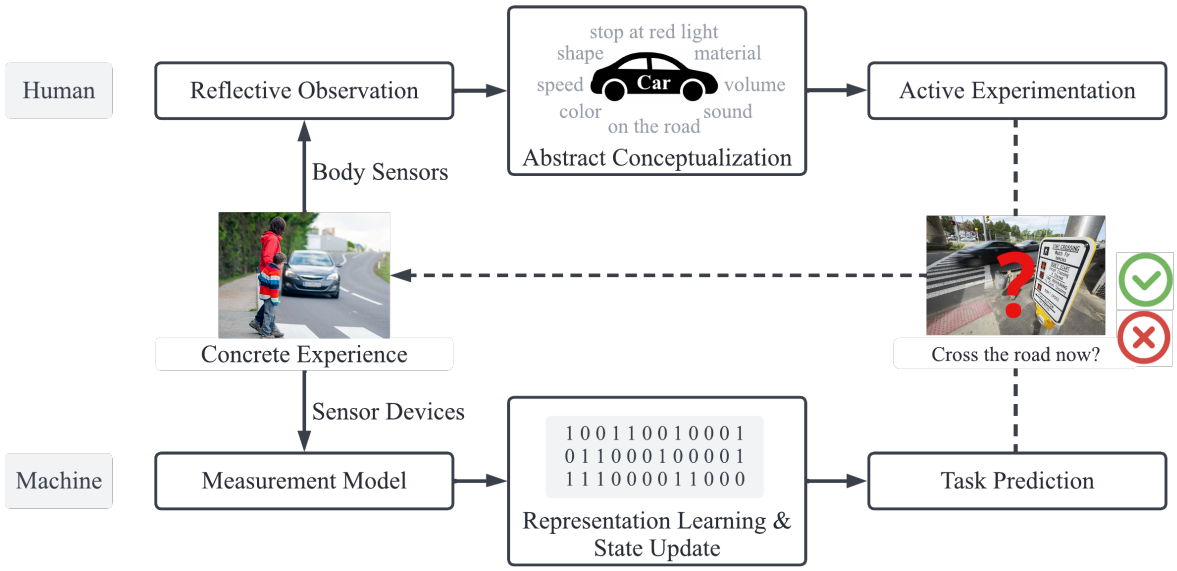

Figure 1.1: Comparison of human and machine learning cycle.

Human interaction with the physical world benefits from the ability to re-use past experiences, both in the short-term, via functions of recent measurements gathered in the current environment, and in the long-term, transferred from other environments. For instance, as illustrated in the Fig. 1.1, imagine how kids learn the concept of "car". They may have concrete experiences of an actual car in the surroundings, from which they perceive various kinds of signals through their body sensors. After exploring the environment more, they start to propose general regulations or assumptions of these objects by summarizing similar

characteristics or distinguishing against the others. Under teachers' instruction, they bind their discoveries with the naming "car" and store it as a piece of knowledge in their long-term memory. In the future, they can validate their cognition by solving practical problems, such as predicting a car motion or deciding whether to cross the road.

Educationalists [Kol84] summarize this learning cycle as a four-stage loop (shown in Fig. 1.1): first be engaged in a concrete experience (CE) where we collect reflective observation (RO), then transfer into abstract conceptualization (AC) and finally validate by active experimentation (AE). While these steps seem natural to humans, in the field of machine learning (ML), it is fairly challenging as we need to cast the steps into machine-executable programs.

Humans are constantly observing new things from their surroundings on-the-fly: eyes take in lights reflected from objects' surfaces, ears take in sound waves of the vibrate, the nose inhales particles from the air, the skin activates on touching, and the tongue distinguishes food moleculars. Despite cutting-edge sensors have been developed to preceive the world, however, mathematical models are still demanded to convert physical or digital signals from different data domains into unified, machine-readable data, which we call *data representations*.

Knowledge is created from the data we memorized. Ideally, our brain should carry all we received, so we do not lose track of any bit of information in order to solve future problems. This is impossible for human to achieve this, given the bounded size of human brain, and in the meantime, unnecessary - to safely across a road, we need to detect cars and understand the traffic lights, yet no need to figure out their model, make or license number. We humans are clever enough to store only *useful* information: the knowledge that is crucial for us to solve practical problems impinging on our survival. Then one may ask: "what information is regarded as useful for machines?" Well, as the machines are created to facilitate humans, the knowledge stored in the machine memory should be valuable enough to achieve the goals set up by humans.

Another question we are interested is: how to apply our learned skills to solve actual problems? The knowledge should be generalized enough to transfer among different situations. For example, the program that tracks an apple falling from the tree is demanded to track a falling orange, even if it has not seen an orange yet. This involves training the machine to capture generic regularities of the physical world, either from direct supervision, by penalizing results different from what we expected, or from the modeling itself, by introducing decent architectures to match desired functionality.

Our ultimate goal is to develop machine learning models to imitate the human learning behavior based on real-world sequential data like videos. In this dissertation, we first introduce the concept of "task-sufficient dynamical model", then show that it is realizable with deep learning architectures, and further suggest feasible methodologies to impose generic priors of the physical world on the learned representation.

1.2 Learning from video data

Although the observations of the world can be collected from different resources, we choose to instantiate the models with optical measurements of videos captured by a variant of cameras, serving to transform the measurements of the light of the scene into digital signals.

Even with modern technologies of fast-speed computation units and high-volume memory devices, modeling sequential video data can be very challenging. First, as videos are high-dimensional in both spatial and temporal horizons, one can easily lose track of long-term memory if there is no generalization from the past data. Hence, videos are noisy and incomplete: they lose one entire dimension in the projection from the 3-D world to the 2-D image, whose quality suffers from numerous failures such as vibrating, blurring, or being out of focus. Thus, the captured video is just a single datum sampled from some probability density, and there is no way for us to approach the "real state" of the world if no proper uncertainty model is established.

However, from another perspective, if we understand how pixels evolve from frame to frame, we should have captured some regularities imposed by the physical world. For example, a cluster of pixels is more likely to move together as they belong to the same object; an object constantly appear and disappear in the video might be occluded by another object closer to the camera. Learning the dynamics of videos is, in essence, the problem of modeling the generic regularities of the scene [MSK04] under certain assumptions or constraints.

1.3 Outline

In Chapter. 2, we first describe a classical way of modeling a dynamic system by Bayesian optimal filter, followed by an introduction of a few sub-optimal filters proposed to approximate the complex posterior probability.

In Chapter. 3, we propose the task-sufficient filtering framework integrating an extra task variable into the filtering framework, that maintains an abstract hidden state sufficient to update itself and predict the task variable. Relation and comparison with the classical Bayesian filtering framework are discussed.

In Chapter. 4, we show that the proposed framework is realizable by recurrent neural networks (RNNs) whose hidden state represents an estimate of the abstract hidden state, and can be supervised by annotated data from the task. We validate the learned representation by producing a maximum a posteriori (MAP) estimate on the task posterior (after bypassing a task prediction model) and comparing on a standard object tracking benchmark.

In Chapter. 5, we show the significance of introducing generic priors, such as topological and regularity constraints imposed by the physical scene, to the modeling. We demonstrate the model's capability of segmenting moving objects from the video frames as video object segmentation problem without supervision from human annotated data.

In Chapter. 6, we analyze the maximum capacity of the learned representation by enforcing it to predict the high-dimensional observation data not yet observed (future data).

The process can be used as a generic proxy task for generating representations for different downstream tasks. We model by introducing hypotheses of partitioning the video frames into semantically-consistent regions and learning the class-specific motion regularities from unlabeled video data.

CHAPTER 2

Modeling the Dynamic System

People use a *stochastic process* to describe the evolution of a random phenomenon against some other variable or set of variables, such as time series. Since the middle of the last century, stochastic processes have been widely used to model mathematical systems in biology, chemistry, neuroscience, physics, control theory, signal processing, etc. Concrete examples include the growth of a bacterial population, the variations in stock prices, or a location of a moving object.

In this chapter, we first review notations and terminologies associated with a stochastic process (Sec. 2.1) and describe general assumptions of the models with unobserved variables (Sec. 2.2). Then we define an optimal filter with classical Bayesian settings (Sec. 2.3), which is realized by several sub-optimal filters (Sec. 2.4) in practice, such as the Kalman filter (KF), particle filter (PF), and grid-based filter (GBF).

2.1 Notations

Each random variable of the stochastic process is uniquely associated with an index such as a timestamp t. A stochastic process of random variable $\boldsymbol{x}$ can be denoted as $\boldsymbol{X}_{t=1}^{T} \doteq \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T\}$,[1] where each $\boldsymbol{x}_t$ represents the random variable at time t. We use the subscript or superscript of $\boldsymbol{X}_{t=1}^{T}$ to indicate the starting and ending timestamp of the series, respectively. In this dissertation, we omit the subscript if the starting timestamp is 1, i.e.,

[1]Note that $\boldsymbol{x}_t$ here is regarded as an abuse of function notation - $\boldsymbol{x}_t$ refers to the random variable indexed by t, not the entire stochastic process.

$\boldsymbol{X}^T \doteq \boldsymbol{X}_{t=1}^T$.

The random variables $\boldsymbol{x}_t$ all take values from the same mathematical space, known as *state space*, which can be either discrete (e.g., finite set of numbers) or continuous (e.g., real-valued numbers). Due to the randomness, a stochastic process can have many outcomes, while a single outcome of the process is called a *sample function* or *realization*.

2.2 Modeling with unobserved variables

In many real-world problems, the variables we wish to estimate are not directly observed but inferable from other observations. For example, the quality of life of a person is generally unobservable, of which the measurements come from measurable variables like wealth, employment, health, etc. To be consistent, we use random variable $\boldsymbol{x}$ to denote unobserved signals (hidden states) and $\boldsymbol{y}$ as the observations of $\boldsymbol{x}$.

Causality. The observations arrive sequentially in time, and one wishes to update the posterior density online, or *casually*. We assume that the system output depends on past and current inputs, not future inputs, i.e., the system state $\boldsymbol{x}_{t+1}$, depends only on inputs $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$. For instance, the model aiming to track the status of an aircraft in real-time takes in sensor data from the radar, which come one datum at a time and only available up to the current time.

Markov property. Per the computation complexity concern, we don't want to store all the past data in order to make current or future prediction. This requires the model to be "memoryless", or *Markovian*, that the conditional probability of future states of the process (conditional on past and present) depends only upon the present state. In other words, we assume $\mathcal{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) = \mathcal{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t)$.

Problem definition with unobserved data. Assuming that the stochastic process is modeled by a random variable $\boldsymbol{x}$, known as the "hidden state". The state cannot be directly observed, but can be monitored through measurements of another random variable $\boldsymbol{y}$ which

is available up to current time t, i.e., $\boldsymbol{Y}^t \doteq \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$. The goal is to build an estimate of the hidden state (e.g., a posterior density) given the observations, either at the current time $\mathcal{P}(\boldsymbol{x}_t \mid \boldsymbol{Y}^t)$ (filtering), or at future times $\mathcal{P}(\boldsymbol{x}_{t+\tau} \mid \boldsymbol{Y}^t)$ for some $\tau > 0$ (prediction).

2.3 Optimal Bayesian filter

Bayesian theorem computes and updates the posterior probability after obtaining new data, based on the conditional probability of the latent variable given the observations, and the prior information (beliefs) about the latent variable itself.

Bayes' theorem. Given the proposition $\boldsymbol{x}$ and its evidence $\boldsymbol{y}$, the conditional probabilities of $\boldsymbol{x}$ given $\boldsymbol{y}$ is true can be computed as:

$$\mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{\mathcal{P}(\boldsymbol{y} \mid \boldsymbol{x})\mathcal{P}(\boldsymbol{x})}{\mathcal{P}(\boldsymbol{y})} \tag{2.1}$$

where $\mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y})$ is the *posterior probability* of $\boldsymbol{x}$ after taking the observations of $\boldsymbol{y}$ into account; $\mathcal{P}(\boldsymbol{x})$ is the *prior probability* which describes one's beliefs about $\boldsymbol{x}$ before data $\boldsymbol{y}$ is obtained; and $\mathcal{P}(\boldsymbol{y} \mid \boldsymbol{x})$ is the *likelihood function*, interpreted as the probabilities of the evidence $\boldsymbol{y}$ given $\boldsymbol{x}$ is true. Essentially, Bayesian theorem allows us to update our prior beliefs $\mathcal{P}(\boldsymbol{x})$ after evidence data $\boldsymbol{y}$ is obtained.

Bayesian filtering. Applying the Bayes' theorem to the modeling of the stochastic process yields Bayesian filtering: With an initial distribution $\mathcal{P}(\boldsymbol{x}_0)$, the unobserved signal (hidden states), $\boldsymbol{X}^T \doteq \left\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T; \boldsymbol{x}_t \in \mathcal{S}\right\}$ is modelled as a Markov process with a transition probability $\mathcal{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t)$, whose observations $\boldsymbol{Y}^T \doteq \left\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\right\}$ are conditionally independent on the process $\boldsymbol{X}^T$ and of marginal distribution (likelihood) $\mathcal{P}(\boldsymbol{y}_t \mid \boldsymbol{x}_t)$.

At any time t, the model aims to estimate the posterior distribution $\mathcal{P}(\boldsymbol{X}^t \mid \boldsymbol{Y}^t)$ given by Bayes' theorem[2]:

$$\mathcal{P}\big(\boldsymbol{X}^t \mid \boldsymbol{Y}^t\big) \propto \mathcal{P}\big(\boldsymbol{Y}^t \mid \boldsymbol{X}^t\big)\mathcal{P}\big(\boldsymbol{X}^t\big) \tag{2.2}$$

[2]We use lowercase t to refer to any timestamps within $0 < t \leq T$, whereas uppercase T refers to the last index of the series. $T \to \infty$ if the sequence is infinite in terms of time.

or recursively and causally as

$$\text{Prediction:} \quad \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^t\big) = \int \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \tag{2.3}$$

$$\text{Update:} \quad \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^{t+1}\big) \propto \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big)\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^t\big) \tag{2.4}$$

where in Eqn. 2.3 we marginalize over $\boldsymbol{x}_t$. We omit the denominator as it is just a normalizing factor.

As such, all the information past observations $\boldsymbol{Y}^t$ contained about the hidden state is in the posterior density $\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$, called "belief state"(BS). This allows people to maintain a stochastic system with finite memory: to update to $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^{t+1}\big)$ after $\boldsymbol{y}_{t+1}$ being observed, one only need to memorize $\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$, given that the transition model $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t\big)$ and the measurement model $\mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big)$ are known beforehand.

2.4 Sub-optimal filters

Since the posterior density function can be very complex, obsessive attempts have been made to obtain approximations of the distributions. We list the most classical methods below.

2.4.1 Kalman filter

The Kalman filter [Kal60] approximates the posterior distribution as a multivariate normal (Gaussian) distribution and keeps track of the mean (or estimate) and variance (or uncertainty) of the hidden state on each dimension over time. With this assumption, the transition and measurement models become closed-form and linear. This character makes it popular in many areas like engineering and econometric applications, especially during the 1960s and 1970s, given the modest computation resources available at that time.

Kalman filter is one of the most efficient approximations of the Bayesian optimal filter, as the transition model, measurement model and belief state in Eqn. 2.3 and Eqn. 2.4 are

all parameterized normal distributions, i.e.,

$$\begin{aligned} \text{Transition model:} \quad & \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t\big) \doteq \mathcal{N}\big(\boldsymbol{A}_{t+1}\boldsymbol{x}_t; \boldsymbol{\Sigma}_{t+1}\big) \\ \text{Measurement model:} \quad & \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big) \doteq \mathcal{N}\big(\boldsymbol{B}_{t+1}\boldsymbol{x}_{t+1}; \boldsymbol{\Gamma}_{t+1}\big) \\ \text{Posterior density:} \quad & \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^{t+1}\big) \doteq \mathcal{N}\big(\hat{\boldsymbol{x}}_{t+1}; \boldsymbol{\Psi}_{t+1}\big) \end{aligned} \tag{2.5}$$

where $\boldsymbol{x}_t \in \mathbb{R}^m$ is the hidden state, $\boldsymbol{y}_t \in \mathbb{R}^n$ the observations and $\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma})$ is a normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The transition model transforms $\boldsymbol{x}_t$ to the next step by a linear matrix $\boldsymbol{A}_{t+1} \in \mathbb{R}^{m\times m}$ and an estimated covariance matrix $\boldsymbol{\Sigma}_{t+1} \in \mathbb{R}^{m\times m}$, the measurement model projects $\boldsymbol{x}_{t+1}$ into the space of observation $\boldsymbol{y}_{t+1}$ by a matrix $\boldsymbol{B}_{t+1} \in \mathbb{R}^{n\times m}$ with an estimated covariance of observation noise $\boldsymbol{\Gamma}_{t+1} \in \mathbb{R}^{n\times n}$. Hence, the posterior density it maintained, is also a normal distribution parameterized by a posteriori state estimate $\hat{\boldsymbol{x}}_{t+1} \in \mathbb{R}^n$ and a covariance estimate $\boldsymbol{\Psi}_{t+1} \in \mathbb{R}^{n\times n}$.

Kalman filter dynamically maintains a state estimate (mean) and its covariance matrix of the Gaussian posterior density in a way that minimizes the mean-square error in the state estimation as $\|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|_2^2$. Theoretically, the Kalman filter is the *optimal filter* if the "true" system is linear, which seldomly holds in real-world problems. Thus, the extended Kalman filter (EKF) is introduced as a non-linear enhancement of the Kalman filter, allowing its state transition and observation model to be non-linear functions. However, unlike the Kalman filter, the EKF is not an optimal estimator in general. In addition, EKFs can quickly diverge if the initial estimation of the state or the process instantiates incorrectly.

2.4.2 Sequential Monte Carlo methods

To not be subject to any linearity or Gaussianity constraints, from the 1990s, considerable efforts have been devoted towards the development of Monte Carlo (MC) integrated methods [Dou98, DGA00, GSS93, Iba01, Laf96]. The core idea is to approximate the intractable posterior density by a large number of samples.

More precisely, at any time step t, assume that we are able to sample N independent

and identically distributed (i.i.d.) random samples $\{\boldsymbol{X}^t[1], \ldots, \boldsymbol{X}^t[N]\}$, called "particles". Equipped with law of large numbers, one wish that with decent amount of particles, the empirical measure:

$$\widehat{\mathcal{P}}_N(\boldsymbol{X}^t \mid \boldsymbol{Y}^t) \doteq \frac{1}{N}\sum_{i=1}^{N} \delta_{\boldsymbol{X}^t[i]}(\boldsymbol{X}^t) \tag{2.6}$$

is able to simulate the posterior of the optimal filter as in Eqn. 2.2. Here $\boldsymbol{X}^t[i]$ denotes the i^{th} particle within time range $[1, t]$ and $\delta_{\boldsymbol{a}}$ stands for the Delta-Dirac mass located in $\boldsymbol{a}$.

Sequential Importance Sampling (SIS). Given that the posterior distribution can be multi-variate and non-standard, however, it is usually impossible to sample efficiently from the posterior distribution with finite number of particles. To tackle this, people introduce *importance sampling* methods, whose core idea is to approximate the density by sampling particles from an arbitrary "support" distribution $\boldsymbol{\pi}$, called *importance sampling distribution*, each with an importance weight. Formally, given the current simulated particles $\{\boldsymbol{X}^t[i]; i = 1, \ldots, N\}$ sampled from the distribution $\boldsymbol{\pi}(\boldsymbol{X}^t \mid \boldsymbol{Y}^t)$, the weighted empirical measure is given by[3]:

$$\widetilde{\mathcal{P}}_N(\boldsymbol{X}^t \mid \boldsymbol{Y}^t) \doteq \sum_{i=1}^{N} \boldsymbol{s}^t[i]\delta_{\boldsymbol{X}^t[i]}(\boldsymbol{X}^t); \text{ where } \boldsymbol{s}^t[i] \doteq \frac{\mathcal{P}(\boldsymbol{X}^t[i] \mid \boldsymbol{Y}^t)}{\boldsymbol{\pi}(\boldsymbol{X}^t[i] \mid \boldsymbol{Y}^t)} \in \mathbb{R}^T \tag{2.7}$$

By injecting into the iterative steps in Eqn. 2.3 and Eqn. 2.4 of optimal Bayesian filter, we can formulate SIS as an iterative process updating the importance weight:

$$s_t[i] \propto s_t[i] \cdot \mathcal{P}(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}[i]) \cdot \mathcal{P}(\boldsymbol{x}_{t+1}[i] \mid \boldsymbol{x}_t[i]) \tag{2.8}$$

where $s_t[i] \in \mathbb{R}^1$, $\mathcal{P}(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}[i])$ can be considered as a likelihood measure located at the i^{th} particle, and $\mathcal{P}(\boldsymbol{x}_{t+1}[i] \mid \boldsymbol{x}_t[i])$ is the transition model of each individual particle.

[3]Here we denote $\widetilde{\mathcal{P}}_N(\cdot)$ as the weighted empirical measure, to distinguish with the unweighted version denoted as $\widehat{\mathcal{P}}_N(\cdot)$.

2.4.3 Particle filter

In practice, a general issue the SIS method encountered is that, as the time increases, the distribution of importance weights $s_t[i]$ becomes more and more skewed, known as the "degeneracy" problem. As such, Gordan et al. [GSS93] proposed the bootstrap particle filter, which in essence is an practical approach to eliminate the particles with low importance weights while duplicate particles with high importance weights.

Pros and Cons. The family of Monte Carlo methods in filtering is very flexible, easy to implement, and parallelizable in many general settings. However, with the thriving of deep neural networks, particle filters become obscure to fit into an end-to-end training schema such as the back-propagation of a neural network. Although a few attempts have been made to cast the particle filtering into deep neural networks [JRB18], they are either computationally inefficient to run on GPUs with batch-wise operator, or requires complicated hand-craft design.

2.4.4 Grid-based filter

Grid-based filters (GBFs) can be regarded as a brute-force sampling strategy in the family of Monte Carlo methods. In a grid-based representation, samples are located at a fixed-size mesh grid with equal intervals, whose importance weights reflect the probability at that location. Compared to other sequential Monte Carlo methods such as particle filters, the grid-based filter also retains the advantage of having no underlying assumption of linearity or Gaussianity, while eliminates the important sampling step at each update, which can be a potential weakness point in the iteration. Even in continuous scenarios, GBFs can produce accurate state estimate as no approximation errors are caused by inappropriate sampling or resampling. However, given limited computation capability in the early years, GBFs were not studied much in the past decades before they were deprecated and replaced by methods with down-sampling (e.g., particle filters) proposed to avoid exponential complexity. Nowadays,

it is worthy for us to re-visit the algorithm of GBFs, as they are naturally compatible with tensor operations of deep neural networks.

CHAPTER 3

Freeing Task Variable from the State

In Chapter. 2, we reviewed the Bayesian optimal filter to model a dynamic system, whose core component is to maintain a posterior density of the hidden state that is sufficient to update itself casually and iteratively as a Markov process. However in practice, the model is intractable when it is realized, unless proper approximation methodologies are integrated to represent the posterior density, the transition probability, and the likelihood function.

Even with a perfect simulation of the probability density, there are problems retained in such a framework. First, the hidden state variable $\boldsymbol{x}$ ceases to have the physical meaning as there is no way to associate it to the "true state" of the system, assuming one exists, absent a supervision mechanism that ties it to the real state, such as a direct observation $\boldsymbol{y}_t \doteq \boldsymbol{x}_t$. Hence, the "true" transition and likelihood models are seldom available in practice. Even if the state space is well-defined or some characters of the underlying distribution (like Gaussianity) are known, there are still infinitely many models and state sequences that can generate the given measurement sequence for any finite length.

What is always well-defined is the notion of a *task variable* - the variable of interest. Here, we free the task variable from the hidden state and demand that the latter be sufficient to update itself and predict the task variable [AS18]. In this chapter, we first derives the recursive task-sufficient filtering steps that involve the state update and prediction along with a task variable $\boldsymbol{z}$ (Sec. 3.2), and then address its relation to the Bayes' filtering (Sec. 3.3 and Sec. 3.4).

3.1 Notations

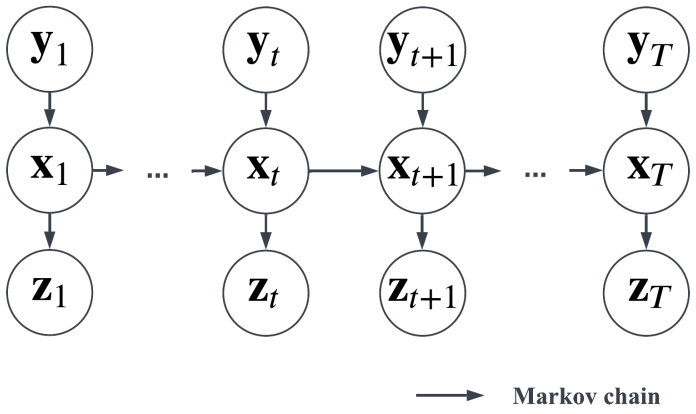

Figure 3.1: Markov chain of hidden state $\boldsymbol{x}$, observations $\boldsymbol{y}$ and task variable $\boldsymbol{z}$.

Task variable. We denote $\boldsymbol{z}_t$ as the quantity of interest (task) at time frame t, and $\boldsymbol{Z}^T \doteq \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_T\}$ as the time series of task up to time T. We omit the subscript if the time series starts with $t = 1$, i.e., $\boldsymbol{Z}^T \doteq \boldsymbol{Z}_{t=1}^T$.

Sufficiency. Given random variable $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$, the posterior density $\mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y})$ is *sufficient* of $\boldsymbol{y}$ for $\boldsymbol{z}$ if:

$$\mathcal{P}(\boldsymbol{z} \mid \boldsymbol{y}) = \int \mathcal{P}(\boldsymbol{z} \mid \boldsymbol{x}) \mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y}) d\boldsymbol{x} \tag{3.1}$$

Hence, we say $\boldsymbol{x}$ is a *sufficient representation*[1] of $\boldsymbol{y}$ for $\boldsymbol{z}$ if they form a Markov chain of $\boldsymbol{y} \to \boldsymbol{x} \to \boldsymbol{z}$, i.e., $\mathcal{P}(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) = \mathcal{P}(\boldsymbol{z} \mid \boldsymbol{y})$. Note that, the posterior distribution of the sufficient representation is always sufficient, as

$$\mathcal{P}(\boldsymbol{z} \mid \boldsymbol{y}) = \int \mathcal{P}(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{y}) d\boldsymbol{x} = \int \mathcal{P}(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{y}) \mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y}) d\boldsymbol{x} = \int \mathcal{P}(\boldsymbol{z} \mid \boldsymbol{z}) \mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y}) d\boldsymbol{x}$$

while the converse does not hold. We abuse the notation "sufficient" to refer to the case when both the variable $\boldsymbol{x}$ is a sufficient representation and the posterior of $\boldsymbol{x}$ is sufficient.

[1]We call $\boldsymbol{x}$ a representation of $\boldsymbol{y}$ if $\boldsymbol{x}$ are sampled by any stochastic function of $\boldsymbol{y}$, i.e., $\boldsymbol{x} \sim \mathcal{P}(\boldsymbol{x} \mid \boldsymbol{y})$. A simple but common case in machine learning is that $\boldsymbol{x}$ is a deterministic function of $\boldsymbol{y}$, $\boldsymbol{x} \doteq \phi(\boldsymbol{y}; \boldsymbol{w})$ from some functions parameterized by $\boldsymbol{w}$.

Fig. 3.1 demonstrates a common case of formulating a sufficient representation in a time series. Denote $\boldsymbol{x}_t$, $\boldsymbol{y}_t$ and $\boldsymbol{z}_t$ the hidden state, observed data and task variable at any time t where $1 \leq t \leq T$, the hidden state $\boldsymbol{x}_t$ is a sufficient representation of all the observed data up to time t as $\boldsymbol{Y}^t$, in order to predict the task variable $\boldsymbol{z}_t$, if no new observation comes.

3.2 Derivation of task-sufficient dynamics

As illustrated in Fig. 3.1, given hidden state $\boldsymbol{x}_t$ with no physical meaning, our goal is to learn a parametric model to predict the task posterior $\mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{x}_t; \boldsymbol{w}_{\text{pred}}\big)$ with parameters $\boldsymbol{w}_{\text{pred}}$, and a state update model $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}; \boldsymbol{w}_{\text{update}}\big)$, to maintain an estimate of the posterior density $\mathcal{P}\big(\boldsymbol{x}; \boldsymbol{w}_{\text{memory}}\big)$ representing the (short-term) memory of the system. For simplicity of notation, we omit the parameters $\boldsymbol{w}$ as it is understood that all functions $\mathcal{P}(\cdot)$ are parameterized.

Then, starting from an initial distribution $\mathcal{P}\big(\boldsymbol{x}_0\big) \doteq \mathcal{P}\big(\boldsymbol{x}_0 \mid \emptyset\big)$, we impose that the learned model be sufficient for task prediction and state update as following.

Task prediction. The hidden state $\boldsymbol{x}_t$ is sufficient of $\boldsymbol{Y}^t$ for the task $\boldsymbol{z}_t$ at time t:

$$\mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{Y}^t\big) = \int \mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{x}_t\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \tag{3.2}$$

This means that the memory $\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$ and the task posterior distribution $\mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{x}_t\big)$ that predicts the task variable given the state, are sufficient to approximate the real posterior density $\mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{Y}^t\big)$ of task given past observations.

State update. The posterior density of $\boldsymbol{x}_t$ is sufficient for $\boldsymbol{x}_{t+1}$:

$$\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^{t+1}\big) = \int \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \tag{3.3}$$

That is, the posterior of the representation $\boldsymbol{x}_t$ contains enough information about the dynamics to update itself given new measurements.

3.3 Relation to Bayesian optimal filter

The classical Bayesian filtering (described in Sec. 2.3) can be considered as a particular case of the above process when the task is to predict the data at the next time. The task-sufficient dynamics is also optimal but does not need to apply Bayes' rule in the iterative process.

Precisely, by applying the Bayes' rule to the state update term $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}\big)$ yields:

$$\begin{aligned}\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{Y}^{t+1}\big) &= \int \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \\ &\propto \int \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{x}_{t+1}\big)\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \\ &\propto \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big) \int \mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t\end{aligned}$$

which is indeed the combination of the iterative state prediction (Eqn. 2.3) and update step (Eqn. 2.4) in Bayesian filtering framework. Note that $\mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{x}_{t+1}\big) = \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big)$ holds with the Markovian property. Hence, by setting the task variable $\boldsymbol{z}_t \doteq \boldsymbol{y}_{t+1}$, Eqn. 3.3 is simply a marginalization over random variable $\boldsymbol{x}_t$:

$$\mathcal{P}\big(\boldsymbol{z}_t \mid \boldsymbol{Y}^t\big) \doteq \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{Y}^t\big) = \int \mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_t\big)\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)d\boldsymbol{x}_t \tag{3.4}$$

3.4 Properties of task-sufficient representation

First, note that we only process data causally, one at a time, yet we obtain an estimate of the posterior given all past measurements. There is no need to carry around, store, or process batches of data.

Also, note that the posterior of the state, $\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$, represents the short-term (running, or ontogenic) memory: it evolves over time and adapts on the fly. Task prediction and state update models, on the other hand, are *static*, as their time-dependence is only through their inputs (current measurement and hidden state). Once learned for a task, they can be transferred to other models. They represent the long-term (phylogenic, transferable) memory of the system. Therefore, our approach naturally breaks down into three components:

a recurrent update, a task-prediction model, and a state transition model, similar to the classical filtering equations.

Like in classical filtering, memory propagation is exact. However, unlike the filtering equations, we can directly learn the transition probability $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}\big)$ without using Bayes' rule, and therefore there is no need to compute the (generative) likelihood function $\mathcal{P}\big(\boldsymbol{y}_{t+1} \mid \boldsymbol{x}_{t+1}\big)$, which is generally intractable for high-dimensional and complex data such as natural images. This is a key innovation of our framework.

Finally, unlike Eqn. 2.4, our model only involves distributions over $\boldsymbol{z}_t$ and $\boldsymbol{x}_t$, which are typically of lower dimension than the data $\boldsymbol{y}_t$, or at least have a simpler distribution (e.g., Gaussian). When the data is generated by a linear system driven by white, zero-mean Gaussian noise with known parameters, and the task is one-step prediction, our model reduces to the Kalman filter, with some subtle differences.[2]

The inference criterion for all the unknowns involved (hidden state and model parameters) is the maximum likelihood. The models are chosen in the parametric function class like deep neural networks, as we describe in the next chapter.

[2]For example, $\mathcal{P}_t\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$ could either be a Gaussian distribution over speed and position (in which case the update rule is trivial) or a Dirac delta over the mean and covariance matrix (in which case the update rule is the Riccati equation, but there are no integrals to be approximated with Monte-Carlo).

CHAPTER 4

Realizing Task-sufficient Dynamics with Deep Neural Networks

In Chapter. 3, we propose a task-sufficient filtering framework that produces a state estimate that is sufficient to update itself and predict a concrete task variable. This chapter proposes a straightforward approach to realize our task-sufficient framework by recurrent neural networks (RNNs), and to validate the learned dynamic representation on the real-time visual object tracking task. The implementation is conceptually simple: it does not require specific nominations of the physical or semantic attributes of the targeted objects. We also show that the proposed architecture can provide a full multi-modal posterior estimate (rather than a point estimate of the object location in common trackers), which is extensible to different circumstances that require a dynamically and casually updated memory with uncertainty.

4.1 Recurrent Neural Networks

To realize the model, we choose recurrent neural networks [RHW86], a powerful function class and natural candidate to represent the memory and its transition. A recurrent neural network (RNN) is a particular type of deep neural network architecture designed to enable connectivity between nodes along a temporal sequence. RNNs accumulate the input information at each time and store the data in a finite, discontinuous internal memory space. Unlike fully-connected feedforward neural networks, RNNs can process arbitrary lengths of input sequences.

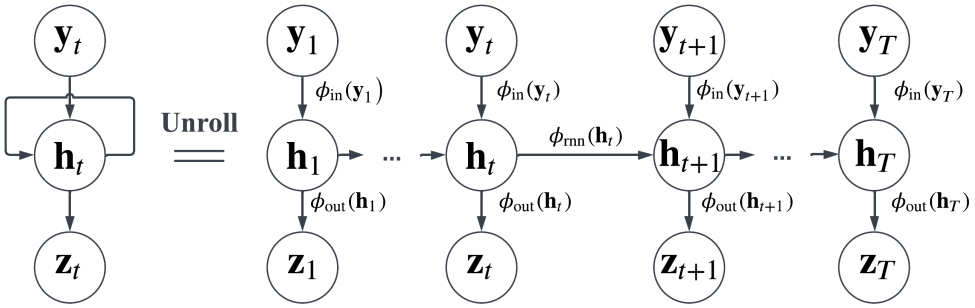

Figure 4.1: Unrolling a recurrent neural network. Here we omit the the parameters $\boldsymbol{w}$ as it is understood that all functions $\phi(\cdot;\boldsymbol{w})$ are parameterized.

Vanilla RNN is a directed acyclic graph that can be unrolled and replaced with a strict feedforward neural network. As shown in Fig. 4.1, the hidden representation of the RNN at each step, denoted by $\boldsymbol{h}_{t+1}$, persists the historical information by applying a transition model ϕ_{rnn} to the previous state, and absorbs the representation of the current-time observation as $\phi_{\text{in}}(\boldsymbol{y}_{t+1})$. Note that we use the notation $\boldsymbol{h}_t$ instead of the generic hidden state variable $\boldsymbol{x}_t$ to specifically refer to the hidden representation of an RNN, which can be considered as an estimate of the hidden state, i.e., $\boldsymbol{h}_t \doteq \widehat{\mathcal{P}}_N(\boldsymbol{X}^t \mid \boldsymbol{Y}^t)$. The network modules, ϕ_{rnn}, ϕ_{in} and ϕ_{out} are all time-invariant, so that the same functions can be deterministically applied to every temporal snapshots, no matter where the node locates or how long the sequence is.

LSTMs, or long short-term memory networks, were first introduced by Hochreiter and Schmidhuber [Gra12] in 1997 as a type of RNNs with special designs to (partially) solve the vanishing gradient problem in implementation: given limited computation precision of the computation units, the gradient values of a vanilla RNN can approach zero or infinity during training, since we can always unroll an RNN into an infinite-length feedforward network whose gradient's precision error can aggregated throughout the long back-propagating chain. To tackle this, LSTMs introduce gates at each time step to selectively "persist" or "forget" information, where each gate is a weighted sum of input data and previous hidden state followed by an activation function.

Convolutional LSTM. Many variants are introduced in implementing the LSTM units, and a typical enhancement is to integrate with the convolution operator into the recursive unitl, called Conv-LSTM [SCW15]:

$$\begin{aligned}
&\text{Forget gate: } \boldsymbol{f}_{t+1} \doteq \boldsymbol{\sigma}\big(\boldsymbol{w}_{\mathrm{f,in}} * \boldsymbol{y}_{t+1} + \boldsymbol{w}_{\mathrm{f,h}} * \boldsymbol{h}_t + \boldsymbol{w}_{\mathrm{f,c}} \circ \boldsymbol{c}_t + \boldsymbol{b}_f\big)\\
&\text{Input gate: } \boldsymbol{i}_{t+1} \doteq \boldsymbol{\sigma}\big(\boldsymbol{w}_{\mathrm{i,in}} * \boldsymbol{y}_{t+1} + \boldsymbol{w}_{\mathrm{i,h}} * \boldsymbol{h}_t + \boldsymbol{w}_{\mathrm{i,c}} \circ \boldsymbol{c}_t + \boldsymbol{b}_i\big)\\
&\text{Cell vector: } \boldsymbol{c}_{t+1} \doteq \boldsymbol{f}_{t+1} \circ \boldsymbol{c}_t + \boldsymbol{i}_{t+1} \circ \tanh\big(\boldsymbol{w}_{\mathrm{c,in}} * \boldsymbol{y}_t + \boldsymbol{w}_{\mathrm{c,h}} * \boldsymbol{h}_t + \boldsymbol{b}_c\big)\\
&\text{Output gate: } \boldsymbol{o}_{t+1} \doteq \boldsymbol{\sigma}\big(\boldsymbol{w}_{\mathrm{o,in}} * \boldsymbol{y}_t + \boldsymbol{w}_{\mathrm{o,h}} * \boldsymbol{h}_t + \boldsymbol{w}_{\mathrm{o,c}} \circ \boldsymbol{c}_{t+1} + \boldsymbol{b}_o\big)\\
&\text{Hidden vector: } \boldsymbol{h}_{t+1} \doteq \boldsymbol{o}_{t+1} \circ \tanh\big(\boldsymbol{c}_{t+1}\big)
\end{aligned} \tag{4.1}$$

where $\boldsymbol{y}_t$ is the input vector (observation) to the LSTM unit; $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ and $\boldsymbol{o}_t \in (0,1)^d$ are d-dimensional activation vectors of the forget gate, the input gate, and the output gate, respectively; $\boldsymbol{h}_t$ and $\boldsymbol{c}_t \in (-1,1)^d$ together represent the hidden state up to time t (i.e., $\widehat{\boldsymbol{X}}^t \doteq \big[\boldsymbol{c}_t, \boldsymbol{h}_t\big]$), called "hidden vector" and "cell vector" respectively; $\boldsymbol{\sigma}(\cdot)$ and $\tanh(\cdot)$ are sigmoid and hyperbolic tangent activation functions; $*$ is the convolution operator; $\circ$ denotes the Hadamard product (element-wise product). The initial state $\hat{\boldsymbol{x}}_0 \doteq \big\{\boldsymbol{c}_0, \boldsymbol{h}_0\big\}$ is set to all zeros.

LSTMs demonstrate significant impacts on multiple disciplines like machine translation, handwriting recognition, and speech recognition, etc., but are less applicable in the domain of computer vision. One possible reason is that, compared to natural language data, image or video data are generally high-dimensional, thus require more computation and memory resource to conduct the training successfully.

In the next section, we will introduce an approach to implement the task-sufficient filtering framework by a convolutional LSTM, whose hidden vector is used to represent a posterior estimate of the hidden state, hence can predict the probability of the task variable conditioned on the data after bypassing a task prediction model.

4.2 Notion of task: generic object tracking

The task-sufficient dynamic framework is general and can be instantiated for many tasks, but we choose to validate on the generic object tracking: Given one sample image of an arbitrary object (target), detect and localize it in all subsequent frames.

Notation. Given a video sequence as observation $\boldsymbol{Y}^T \doteq \{\boldsymbol{y}_0, \dots, \boldsymbol{y}_T\}$, where each frame $\boldsymbol{y}_t \in \mathbb{R}^{H \times W}$, the goal is to predict (the posterior density of) the task variable $\boldsymbol{z}_t$ at each subsequent time. The physical meaning of the task variable varies, for instance, $\boldsymbol{z}_t \in \mathbb{R}^5 \doteq [c, p_x, p_y, o_w, o_h]$ as an object's index c and the scaled translation group for axis-aligned bounding boxes, where the translation component determines the center (p_x, p_y), and the size (o_w, o_h) of the bounding boxes. More in general one could consider the affine or projective group, or general deformations where $\boldsymbol{z}_t \in \{0,1\}^{H \times W}$ is a binary segmentation mask. The hidden state $\boldsymbol{x}_t$ represents whatever internal representation necessary to predict the task variable, and to update the model. Supervision consists of a single sample bounding box $\boldsymbol{z}_0^* \doteq \boldsymbol{z}_0(\boldsymbol{y}_0)$ in the initial frame $\boldsymbol{y}_0$,[1] so that the initial posterior density is:

$$\mathcal{P}(\boldsymbol{z}_0 \mid \boldsymbol{y}_0) \doteq \delta_{\boldsymbol{z}_0^*}(\boldsymbol{z}_0) \tag{4.2}$$

where $\delta_{\boldsymbol{z}_0^*}$ is the Dirac delta function at position defined by $\boldsymbol{z}_0^*$.

Recent tracking schemes have attempted to infer a model (system identification) along with a point estimate[2] of the target pose in subsequent frames, exploiting temporal regularity and complex heuristics for data association [KMM12, HGS16] that often result in trackers that *drift* and *lock* to persistent portions of the background rather than the object of interest.

[1] We abuse the notation and write $\boldsymbol{z}_t(\boldsymbol{y}_t)$ to indicate the video frame $\boldsymbol{y}_t$ restricted to a point estimate $\boldsymbol{z}_t$ at time t, for instance an image patch cropped by a rectangular bounding box. Typically, we use $\boldsymbol{z}_t^*$ to denote a ground truth annotation, while use $\hat{\boldsymbol{z}}_t$ to represent an estimated location.

[2] A point estimate is one particular state – which is generally multi-dimensional – in contrast with a distributional estimate, also known as "belief state", which is the posterior density. This does not mean that the state is a point on the image plane; it could be the parameters of a reference frame or bounding box attached to the object.

Probabilistic inference trackers typically represent states of objects as a distribution with uncertainty. Various kinds of probabilistic inference models have been applied to trackers, including Kalman Filter [Rei79, RSL11], Extended Kalman Filter [ML11] and Particle Filter [KBD04, BRL09, CFL06, YDD05, HF09], the latter when the posterior is assumed to be multi-modal. Other less-popular variants include Exponential Filters, Sum-of-Gaussian filters, and various numerical integration schemes for the optimal filter that tend not to scale to high-dimensional state spaces.

CNN-based trackers generally model tracking as a similarity learning problem. [SMG17] reformulates the classical DCF framework as a one-layer convolutional neural network. By extracting and comparing pre-trained CNN representations of the target image patch and the candidate patch [WY13, HYK15, WOW15, MHY15, DRK16], the real-time computation can be achieved with none or little online training. Other losses have been employed by [SBC, ZGH16, HSA17, TGS16, SBC17] tailored to Siamese or triplet networks. Many end-to-end training trackers either formulate the tracking problem as regression problem [HTS16] or classification problem [NH16]; [BVH16] is the pioneering work for fully convolutional Siamese network and there are a large number of follow-up works, e.g., [HLT18, FPZ17, LYW18]; [HLR17, SR17, CKL17] attempt to learn decision-making policies including feature selection and re-initialization actions; [CCY, COL17, WTX18] use attentional networks.

RNN-based trackers include end-to-end trainable recurrent neural models for multiple purposes. [KMM15, NZH17, KBP17] use RNN to focus computation on task-related information by presenting attention module or spatial supervision;[ZMW17] formulates an RNN agent that can be trained with reinforcement learning to capture inter-frame correlations. RNNs are natural candidates for accumulating temporal information in tracking. Many works [SAS17, GFZ17, YC17, GFF18, KLR18] explored the potential of RNN to perform appearance feature embedding of the target. There are a few attempts in using RNN to reformulate filtering or prediction. [MRD17] use customized RNNs to cast the classical Bayesian state estimation and data association under the track-by-detection framework. [XGY17] use

LSTM to predict future actions of cars in the driving dataset.

4.3 Math formulation and toy experiments

Rather than picking a plain RNN architecture, we wish to impose some constraints from the problem domain. Deep convolutional networks trained with the cross-entropy loss approximate, at their penultimate layer, produces the likelihood of the classes given the current image and a region hypothesis. They are, therefore, the natural candidate for a component of our model. A second component exploits short-term regularities that are captured to first-order by correlation. Finally, a recurrent network with finite memory imposes a Markov structure without requiring an explicit model. These choices inform the design of our architecture that captures the complex and multi-modal posterior estimates of the task variable; in tracking, this is the pose of the projection of objects onto the image.

More specifically, we would like to build a recurrent network (RNN) that which implements the transition probability $\mathcal{P}\big(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1}\big)$ (short-term memory), and a decoder network (DNN) that realizes $\mathcal{P}\big(\boldsymbol{z}_{t+\tau} \mid \boldsymbol{Y}^t\big)$ (long-term memory), from which the task memory $\mathcal{P}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$ can be updated via a deterministic integrator. Following the sampling schema of Grid-based Filters (described in Sec. 2.4.4), we approximate the posterior distribution by sampling from a mesh grid defined within a local window of the image domain, called the region-of-interest (ROI):

$$\mathcal{P}_{M\times N}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big) \doteq \frac{1}{M \times N} \sum_{i=1}^{M\times N} s_t[i] \cdot \delta_{\boldsymbol{x}_t[i]}\big(\boldsymbol{x}_t\big) \tag{4.3}$$

where i indexes the sampling point on a 2-D mesh grid with size $M \times N$; $s_t[i] \in \mathbb{R}^1$ denotes the (posterior) density at the location of the i^{th} sample $\boldsymbol{x}_t[i]$; and $\delta_{\boldsymbol{x}_t[i]}(\cdot)$ is the Dirac delta function at the i^{th} sample point. Similar grid-based sampling strategies also apply to approximate the likelihood function of observed data as $\text{Sim}\big(\boldsymbol{y}_t; \boldsymbol{z}_0^*\big)$ (Sec. 4.4.1), the state update probability $\mathcal{P}(\boldsymbol{x}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{y}_{t+1})$ (Sec. 4.4.2), as well as the posterior of the task variable

$\mathcal{P}\left(\boldsymbol{z}_t \mid \boldsymbol{Y}^t\right)$ (Sec. 4.4.3), as described next.

Toy experiments of single and multiple object(s) tracking at pixel-level are demonstrated in Fig. 4.2 and Fig. 4.3.

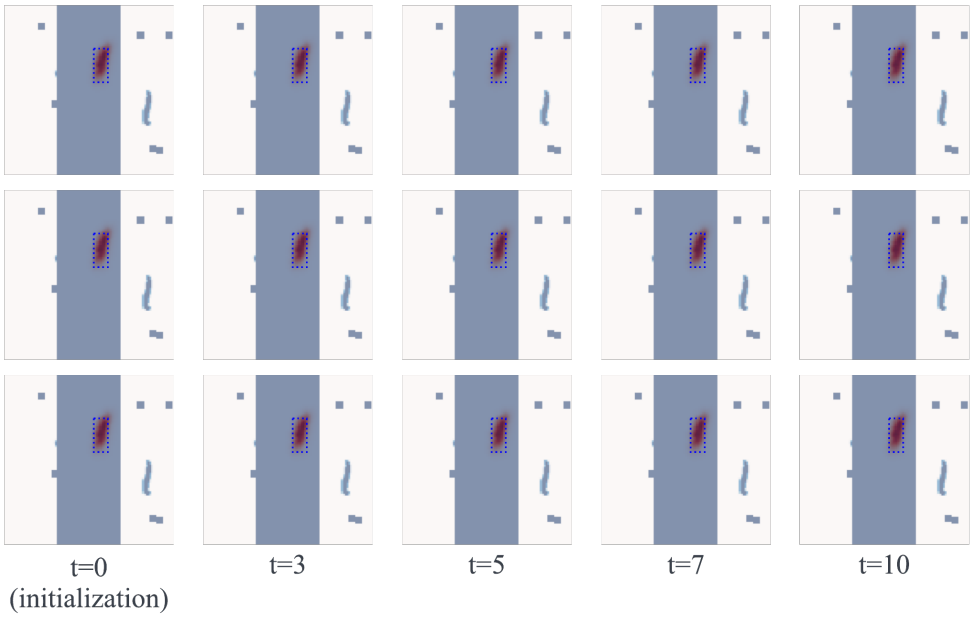

Figure 4.2: Moving MNIST digits undergoing occlusion (central vertical bar in each image). The blue rectangle is the ground truth bounding box. In red is the posterior probability inferred by the network that a given pixel belongs to the digit (exaggerated for clarity).

In Fig. 4.2, we show an illustrative experiment where the task is pixel-level localization (or video segmentation). The network is initialized with the position of the digit in the first frame with the shape unknown. In the top row of Fig. 4.2, the network keeps track of the digit while it is occluded and remembers the approximate shape of the digit. When two similar digits cross, the network keeps tracking. In the central row, the object to track is initially occluded, and its initial velocity is unknown. The network evolves a complex posterior over the position of the object. When the object becomes visible again, the posterior collapses on the object. And in the bottom row, again, the object is initially occluded. When two plausible objects become unoccluded, the network keeps tracking both of them using a multi-

modal posterior rather than collapsing on only one of them.

Note that the initial shape of the target is unknown and becomes increasingly better defined as the sequence goes on, and then more information is accumulated. Meanwhile, a simplistic multi-tracker can be implemented naively using the realization just described by assuming that each object exists and moves independently of others (a patently wrong assumption) and evolving independent representations for each. With the similar setting as Fig. 4.2, in Fig. 4.3, we model the hidden state as a joint state estimation of multiple objects and assume independent task posteriors for each target. The number of targets is known, while the shape is unknown. The posterior correctly converges to two modes. At the top row, the network correctly tracks the top "5" and remembers the approximate shape when being occluded. The network also correctly collapses on the bottom "5", initially partially occluded. And at the bottom row, two targets are both initially occluded. The network disperses the probability with large uncertainty until the digits are detected.

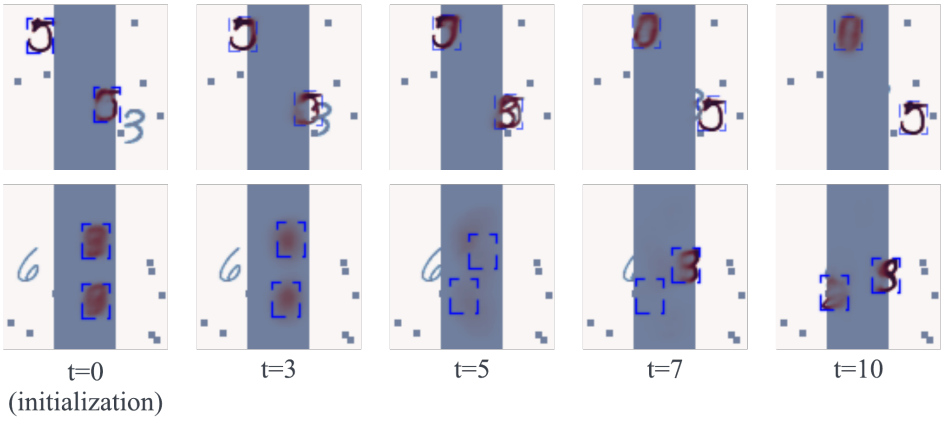

Figure 4.3: Sample experiments on multi-modal tracking with the same setting as Fig. 4.2.

Also note that, the evolution being deterministic and the density unimodal, does *not* mean that the task posterior (object location) is. Fig. 4.2 shows the latter is typically multi-modal, reflecting multiple possible locations and trajectories for the objects of interest, even if their dynamics are relatively simple. Note also that, even though images are processed

one at a time, the hidden state represents the cumulative memory. This is illustrated in Fig. 4.2, where a target initially partially occluded becomes visible during the sequence. Once fully seen, it remains known (and its position predicted) even when fully occluded.

4.4 Methodology

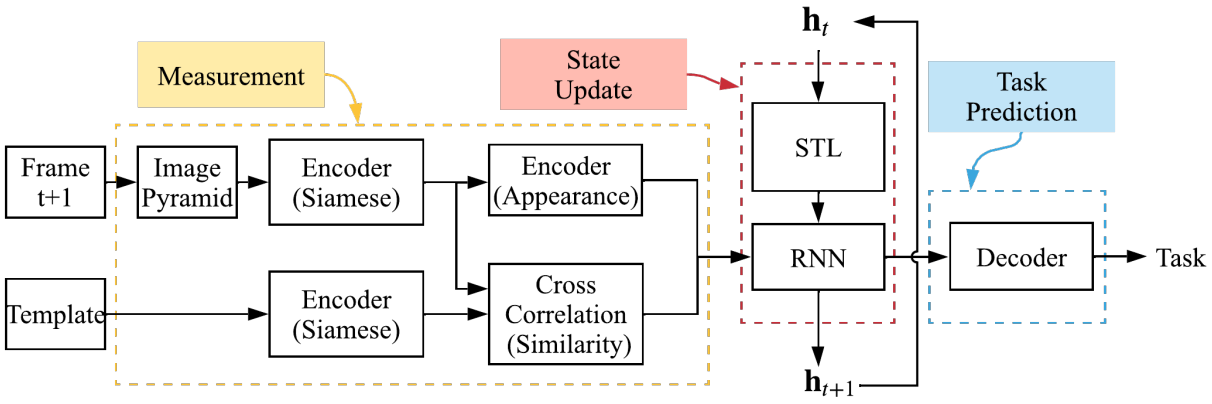

Figure 4.4: Flow-chart of our model at one temporal snapshot. $\boldsymbol{h}_t$ typically refers to the hidden representation of a conv-LSTM.

The overall architecture is illustrated in Fig. 4.4. The previous hidden state (left) is processed, together with the template and ROI of the current frame (top), to yield the updated state (right) and task prediction (bottom).

4.4.1 Measurement model

In addition to long and short-term memory, we retain the initial annotation (or the "template") as it *defines* the task. The resulting architecture uses convolutional layers computed at multiple scales to capture the nuisance variability to which the data is subject, and a correlation layer to relate the resulting representation to the task variable. Note that the correlation layer is not adapted online, so that a tighter connection should be established.

More precisely, the multi-scale similarity representation is generated by taking feature maps of both the template of the object $\boldsymbol{z}_0^*$ and the current image frame $\boldsymbol{y}_t$ extracted by a shared-weighted descriptor parameterized by $\boldsymbol{w}_{\text{siam}}$ as input, and the similarity measurement learns a function $\mathbf{Sim}\big(\boldsymbol{y}_t, \boldsymbol{z}_0^*\big)$ that matches the search area into a similarity density as:

$$\mathbf{Sim}\big(\boldsymbol{y}_t, \boldsymbol{z}_0^*\big) \doteq \phi\big(\boldsymbol{y}_t; \boldsymbol{w}_{\text{siam}}\big) * \phi\big(\boldsymbol{z}_0^*; \boldsymbol{w}_{\text{siam}}\big) + \boldsymbol{b} \tag{4.4}$$

where $\phi\big(\cdot; \boldsymbol{w}_{\text{siam}}\big)$ is the mapping from an image patch to the feature space (activation) specified by the network parameters $\boldsymbol{w}_{\text{siam}}$; $*$ denotes the (normalized) 2-D spatial cross-correlation. In our experiment, similar to [BVH16], we integrate a pre-trained AlexNet [KSH12] architecture as Siamese branches and fine-tune end-to-end with other parts of the network. In practice, the similarity map is also approximated by the grid-based sampling strategy as:

$$\widehat{\mathbf{Sim}}_{M\times N}\big(\boldsymbol{y}_t, \boldsymbol{z}_0^*\big) \doteq \frac{1}{M \times N} \sum_{i=1}^{M\times N} \text{sim}_t[i] \cdot \delta_{\boldsymbol{y}_t[i]}\big(\boldsymbol{y}_t\big) \tag{4.5}$$

Here we use $\text{sim}_t[i] \in \mathbb{R}^1$ to represent the similarity score on the i^{th} sample. Note that $\text{sim}_t[i]$ and $s_t[i]$ are different scalars but can be defined on a same location (i.e., $\boldsymbol{y}_t[i] \doteq \boldsymbol{x}_t[i]$).

The measurement model outputs a concatenation of the similarity score map and the search area's feature vector, yielding:

$$\phi_{\text{meas}}\big(\boldsymbol{y}_t, \boldsymbol{z}_0^*\big) \doteq \Big[\widehat{\mathbf{Sim}}_{M\times N}\big(\boldsymbol{y}_t, \boldsymbol{z}_0^*\big), \phi\big(\boldsymbol{y}_t; \boldsymbol{w}_{\text{siam}}\big)\Big] \tag{4.6}$$

where $[\cdot, \cdot]$ denotes the concatenation operation.

4.4.2 State transition model

State transition model is realized by a recursive unit ϕ_{rnn} of an RNN that takes in the measurement model's output $\phi_{\text{meas}}\big(\boldsymbol{y}_{t+1}, \boldsymbol{z}_0^*\big)$ as well as the current internal memory of RNN as $\boldsymbol{h}_t \doteq \widehat{\mathcal{P}}_{M\times N}\big(\boldsymbol{x}_t \mid \boldsymbol{Y}^t\big)$ and produce the updated hidden representation at $t+1$:

$$\boldsymbol{h}_{t+1} \doteq \phi_{\text{rnn}}\Big(\boldsymbol{h}_t, \boldsymbol{x}_t, \phi_{\text{meas}}\big(\boldsymbol{y}_{t+1}, \boldsymbol{z}_0^*\big)\Big) \tag{4.7}$$

We choose a convolutional long-short-time memory network (Conv-LSTM) [XCW15] as the recurrent unit. It extends a fully connected (FC-LSTM) with convolutional input-to-state and state-to-state transitions structures. This architecture retains fully-convolutional characteristic that accepts inputs of arbitrary size. It also retains spatial relationships among elements of the hidden state $\boldsymbol{x}_t$, which enables the entire system to adapt to the movement of the search region.

4.4.3 Task prediction model

The task prediction model serves to transfer the posterior distribution defined on the hidden state space to the space of task variable. The module consists of a few de-convolutional layers followed by a sigmoid layer, yielding an activation tensor as a discretized grid-based approximation of the task posterior $\mathcal{P}_{H\times W}(\boldsymbol{z}_t|\boldsymbol{Y}^t)$. Note that the mesh grid we used to sample the task posterior is comparable with the image size $H \times W$.

4.4.4 Training loss

Instead of directly comparing with the ground truth data point $\boldsymbol{z}_t^*$, we match the predicted task posterior $\mathcal{P}_{H\times W}(\boldsymbol{z}_t|\boldsymbol{Y}^t)$ with a (pseudo) ground truth heatmap $\widehat{\mathcal{P}}^*_{H\times W}(\boldsymbol{z}_t \mid \boldsymbol{Y}^t)$ generated by computing the Intersection-over-Union (IoU) with each sample $\boldsymbol{z}_t[i]$, and binarize the IoU scores with a threshold, yielding:

$$\widehat{\mathcal{P}}^*_{H\times W}(\boldsymbol{z}_t \mid \boldsymbol{Y}^t) \doteq \sum_{i=1}^{H\times W} \mathbb{1}\Big[\mathrm{IoU}\big(\boldsymbol{z}_t[i], \boldsymbol{z}_t^*\big) > \lambda\Big] \cdot \delta_{\boldsymbol{z}_t[i]}\big(\boldsymbol{z}_t\big) \tag{4.8}$$

with $\lambda = 0.7$. $\mathbb{1}[\cdot]$ is the indicator function, IoU refers to the Intersection-over-Union operation defined as $\mathrm{IoU}(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a} \cap \boldsymbol{b})/(\boldsymbol{a} \cup \boldsymbol{b})$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are bounding boxes. The network is trained using truncated back-propagation through time (TBPTT) [Elm90, WP90] with stochastic gradient descent (SGD) to minimize the binary cross-entropy (logistic loss) as customary.

4.5 Experiments

4.5.1 Engineering concerns

We discuss a few engineering concerns specific to the problem as below.

Search region. To reduce computation complexity, we crop a region-of-interest (ROI) around the mode of the posterior distribution $\widehat{\mathcal{P}}_{H\times W}(\boldsymbol{z}_t|\boldsymbol{Y}^t)$ from the current frame. To estimate translation and scale jointly, we also formulate an image pyramid by adding context margins of different sizes and then apply a set of resizing ratios.

Spatial transformer layer. Cropping ROIs requires transforming the hidden representation in a co-variant manner. We take off-the-shelf spatial transformer layers (STL) [JSZ15] to transform the hidden representation of the RNN according to the predicted motion.

Synthesized data for training. Inspired by [MRD17], we train the network using a combination of actual and synthesized data, the latter obtained by sampling from a simple generative trajectory model learned from real data. We first learn a trajectory model by estimating the mean and variance of the average velocity from annotated trajectories in the sequence. Then we generate tracks by sampling from a normal distribution with the learned parameters. Finally, we render synthesized video frames by randomly selecting background pixels from the same video and then pasting the foreground (randomly extracted from annotated bounding boxes) onto sampled background area based on the generated tracks. We also augment the data by placing random obstacles to the video frames to simulate occlusion and dynamic objects to simulate distractors.

4.5.2 Training

The training uses ALOV300+ [SCC14] with 214 sequences, of which 23 were removed as they overlap with the test set for VOT2017 [KML16]. Although the network can be adapted to arbitrary input sizes, we fixed the ROI to 225×225 and rescaled the template to 127×127,

which we know will incur some performance loss, but at the benefit of inference speed. Due to padding issues, the predicted task posterior is of resolution 199×199, slightly smaller than the original ROI. The RNN is trained using TBPTT [Elm90]: after 10 forward steps we back-propagate 20 steps. The hidden state is of dimension $49 \times 49 \times 16$. We start with a pre-trained AlexNet [KSH12] fine-tuned by [BVH16] as the Siamese backbone while randomly initializing the rest of the network. We fixe parameters up to Conv4 layer and create two Conv5 layers, one to feed into the cross-correlation layer and another to compute the visual representation of the search area of $\boldsymbol{y}_t$. Spatial transformation layers are jointly trained end-to-end. The initial learning rate is 0.0004.

4.5.3 Evaluation

In order to evaluate our method, we use tracking benchmarks, that however comes with some challenges: all existing benchmarks only score point estimates, not the entire state. Therefore, to submit to any of the evaluation benchmarks, we reduce our posterior density to a point estimate, which is an additional aspect of the problem that we do not focus on in this work. This can be as simple as the maximum a posteriori (MAP) estimate (the instantaneous peak of the posterior) or as complex as an explicit model to maintain multiple hypotheses [BL95]. We choose the former, maximum a posteriori, cognizant that in cases where there are multiple modes of the posterior, the MAP estimate can jump between them. Even if this is penalized in the benchmarks, we believe reporting the full posterior is better than any point estimate, as it allows more sophisticated downstream processing when the output is used for decision or control. This is further discussed in Sec. 4.6.

Dataset. We choose the standard object tracking benchmarkVOT2017 [KML16] to evaluate our learned representation. VOT2017 comprises 60 video sequences of various objects in challenging backgrounds. We use the official evaluation toolkit for quantitative evaluation, which produces a combined measure of rankings and scores, along with expected average overlap between predictions and ground truth bounding boxes. The protocol resets

after 5 frames in case of failure, reporting zero accuracies and expected overlap within these frames. Our tracker does not need re-initialization, so it is penalized by this protocol.

Evaluation set-up. We evaluate our methods using a single NVIDIA GeForce GTX 1080 Ti and Intel Core i7-7700K CPU at 4.2GHz. To handle multiple scales, we construct an image pyramid with 3 scales $\left\{1.1^{-1}, 1, 1.1\right\}$, and let the task prediction module learn to regress the posterior at 9 scales $\left\{1.1^{-1}, 1, 1.1\right\} \times \left\{1.1^{-1}, 1, 1.1\right\}$ to handle various aspect-ratios. We crop the template based on the smallest enclosing bounding box to further reduce computation cost, so its size is not fixed during inference.

Evaluation results. We compare our method against a collection of participants of the VOT 2017 challenges. Our quantitative results are reported in Tab. 4.1 using the expected average overlap (EAO) criteria. Note that, we improved 3.9% from SiamFC [BVH16], although the latter one cannot be considered a superficial ablation of our model[3]. Fig. 4.5 shows representative examples of our results against competitive trackers during occlusion, specifically when the object is occluded, our competitors without memory drift to other objects, e.g., another person, human arm, or the trophy in Fig. 4.5. We also discover a phenomena that potentially penalizes our overall accuracy performance: when the object is partially occluded, our trackers report the 2-D bounding boxes projected from 3-D location, while both ground truth annotation and results of many trackers shift to the bounding box enclosing the revealing part of the object.

Ablation studies are performed to quantify the impact of the LSTM. We run the tracker under the same parameters settings while reporting the estimation with the highest response after the measurement module discussed in Sec. 4.4.1 (tagged as "ablation" in Tab. 4.1). Although comparable results are reported, the memory module improves the EAO performance with trivially extra execution cost. Fig. 4.6 shows representative examples of

[3]As discussed in Sec. 4.4.1, to test the power of learned dynamics, we eliminate the post-processing part from [BVH16] which applies a spatial cosine-window penalty to locations from the central location in the heatmap. We also eliminate the temporal aggregation of template representations.

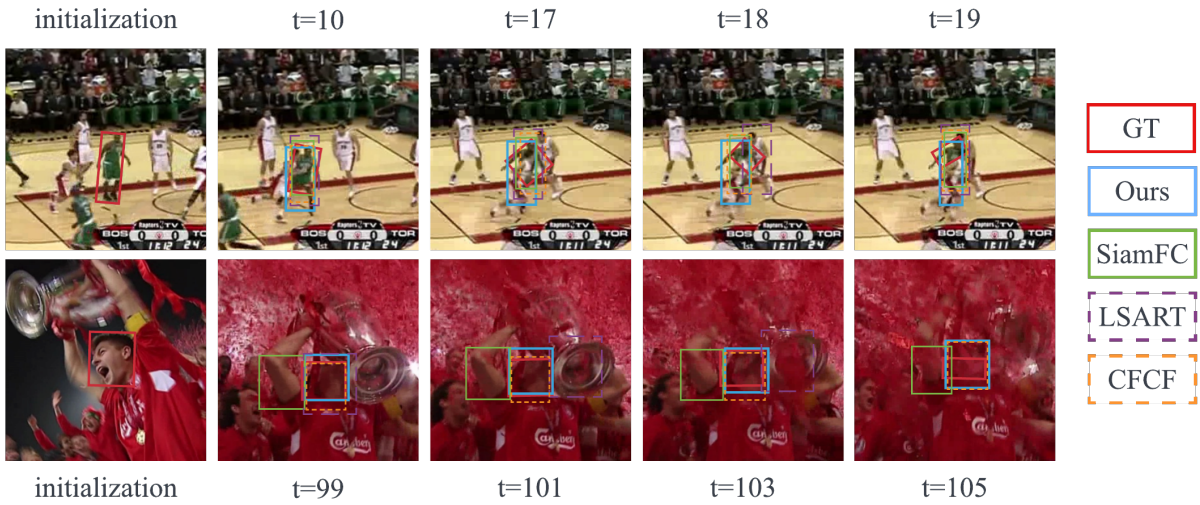

Figure 4.5: A zoom-in visualization of ground truth annotation and the results of our tracker against competitive trackers in occlusion scenario. The left-most image of each line defines the target objects.

both outputs.

4.6 Discussion on predicting the full posterior

Our method produces the entire posterior density of the task variable conditioned on all past data. Alas, there is no public benchmark to evaluate posterior density estimates. To adapt our method to available benchmarks, we need to produce a point estimate. We choose the mode, according to a maximum a-posteriori criterion. We do not implement any post-processing, cognizant that the forced choice of a point estimate may penalize our results in the presence of clutter, where the mode estimate can jump around. The problem is manifest, especially when the evaluation protocol has quirks. For instance, in VOT challenges [KML16], the evaluator forces the tracker to re-initialize 5 frames after a failure is detected. If the true posterior has two modes as of two similar targets, the MAP can jump, thus triggering a "failure", yet the multi-modal estimate of the posterior is smooth and correctly

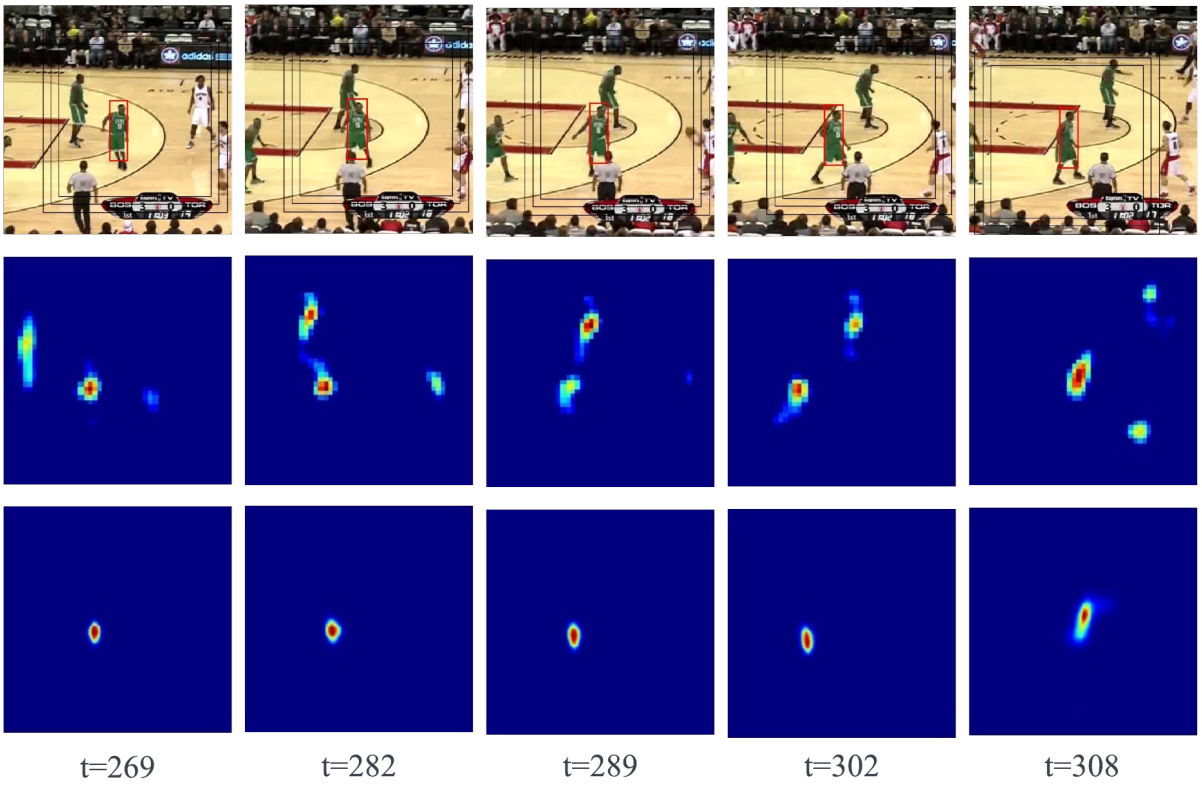

Figure 4.6: A visualization of a subsequence where a distractor enters and leaves the search region while the target is partially occluded. Best viewed on a color display. **Top row** is zoom-in visualization of video frames, ROIs (in black rectangles), and MAP estimate of the posterior (in red rectangles). **Central row** is a slice of heatmaps generated by a cross-correlation layer from one scale. **Bottom row** shows a slice of output posterior from the same scale. The redder the color, the higher the probability (or likelihood). The memory module successfully disambiguates the distractors.

	EAO ↑	FPS ↑	OCC ↑
struck [HGS16]	0.0966	16.9176	0.3635
SRDCF [DHS15]	0.1189	2.4624	0.4194
CGS [DWQ17]	0.1405	0.3906	0.4392
DPT [LZK17]	0.1576	20.7689	0.4156
HMMTxD [VMN16]	0.1682	3.5507	0.4348
SAPKLTF [VM17]	0.1836	31.4607	0.4435
SiamFC [BVH16]	0.1880	31.9134	0.4158
Ours	**0.1955**	21.5630	**0.4592**
Ours (ablation)	0.1840	**32.8331**	0.4132

Table 4.1: Evaluation Results on VOT 2017 benchmark. "EAO" is short for expected average overlap, which means the expected no-reset overlap of a tracker run on a shortterm sequence. "OCC" is short for tag_occlusion.

predicts the presence of two equiprobable targets. Unlike memoryless trackers or point-estimators, our model maintains multiple hypotheses, and smoothly evolves the posterior distribution (Fig. 4.2).

Since our tracker maintains the full multi-modal posterior, updated smoothly over time, "lost tracking" in our case only consists of another region of state-space having a temporarily higher posterior. There is no discontinuity in the estimate of the posterior, just in the location of the maximum. As shown in Fig. 4.7, the tracker obtains the dynamics from the ant, which is still for a few frames and then flicks. A temporal multi-modal posterior is generated with large uncertainty, and the peak of the posterior jumps between two modes. This phenomenon also occurs after object sticks for a long time followed by a sudden move or an unpredictable camera motion. Since we do not include any sophistication in the choice of a point estimate, we observe that in a few sequences, the peak of the posterior jumps between two modes, each time triggering a miss in the evaluation. Even after re-initialization, jumping can occur, thus resulting in perceived multiple misses in the evaluation scheme.

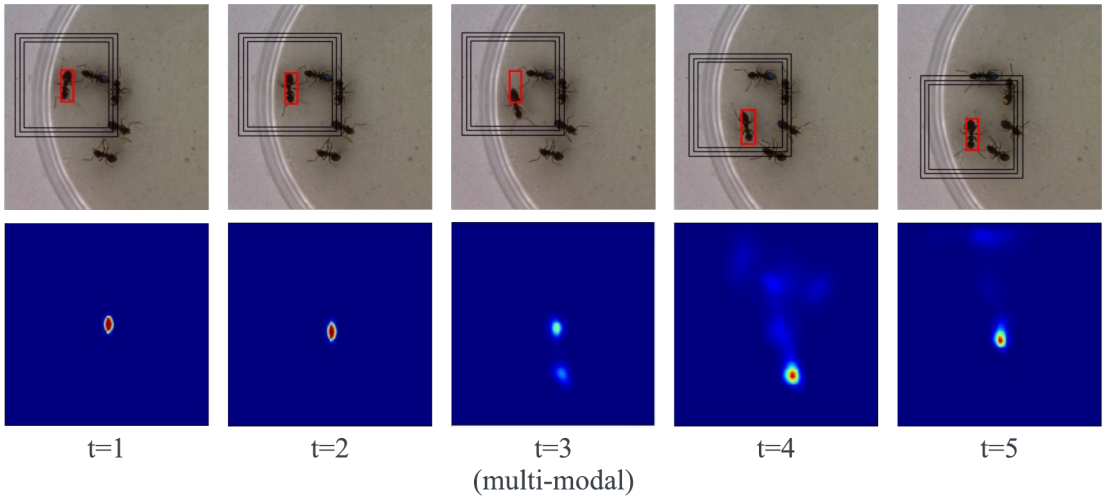

Figure 4.7: A visualization of temporal lose track. **Top row** is zoom-in visualization of video frames, multi-scale ROIs (black rectangles), and MAP estimation of posterior (red rectangles). **Bottom row** is a slice of posterior from one scale.

A more sophisticated logic to extract a point estimate would easily avoid this phenomenon, but is beyond our scope here. The purpose of these experiments is to ascertain whether, even in the presence of a naive choice of the point estimate, our method shows state-of-the-art performance with the simple choice of MAP. Of course, our method provides a full posterior, which offers significant advantages when the tracker is integrated into more complex systems that can ingest, or even require, uncertainty estimates or maintain multiple hypotheses for each tracked object.

CHAPTER 5

Building Object Hypotheses with Generic Prior

In Chapter. 4, we implemented an recurrent neural network as a practical solution to separate the memory "state" – the function of past measurements that is sufficient to predict future ones – from the task variable. Via validation on the standard benchmark, we show that the framework can learn video's dynamical representation that are sufficient to update and predict the task, with no specific modeling of the state itself.

However, there are still problems remaining. First, the representation we learned is used to solve the object tracking problem whose task domain is of low dimension (i.e., the 4-D bounding box and the object index), while the same architecture may under-fit if we apply it to other fine-grained problems such as pixel-level object tracking. Second, collecting data on video-related problems is costly. Third, even though we have sufficient data for training, we may still encounter objects that do not occur in the training set.

In this chapter, we propose an unsupervised method to obtain object-level representation by imposing the generic priors of the scene. The task we choose is detecting and tracking generic moving objects at pixel-level, known as "video object segmentation". To detect and maintain the state of objects efficiently, we leverage generic priors such as temporal consistency, regularity of the resulting deformations, visibility relations, and other Gestalts.

For us, "objects" are regions of 3-D space that project onto simply-connected portions of the image domain, although we abuse the term to refer to such portions of the image instead. We wish to capture the shape and deformation of such image regions, which are informative of characteristics of the objects in 3-D, a process which we refer to as "tracking", as opposed

to simple bounding boxes. We specifically focus on moving objects, whereby relative motion between them and the viewer causes changes in visibility (occlusion).

Our objects do not have names: We do not use a supervised training set, and we do not require manual initialization. Indeed, we wish to assume as little as possible other than the generic regularities arising from physical constraints. One could use our objects to train or initialize a semantic detector.

5.1 Notion of task: video object segmentation

We aim to model the video signals by organizing a video into regions informed by generic priors in the scene, a problem known as "video object segmentation". The goal is to detect and track the generic moving objects in videos.

Notation. Given a video sequence as observations of the scene as $\boldsymbol{Y}^T \doteq \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_T\}$ with each element a video frame $\boldsymbol{y}_t \in \mathbb{R}^{H \times W}$, our goal is to partition the video frame into regions, i.e., $\boldsymbol{y}_t = \bigcup_{i=0}^{N_t} \boldsymbol{r}_t[i]$, corresponding to the background region $\boldsymbol{r}_t[0]$ and a collection of (projections of partially) visible objects $\{\boldsymbol{r}_t[1], \dots, \boldsymbol{r}_t[N_t]\}$ at the current frame t. Note the number of foreground objects N_t can vary over time. We call a point estimate of these regions based on prior image measurements the "object state", i.e., $\hat{\boldsymbol{x}}_t[i] \doteq \boldsymbol{r}_t[i]$, which is maintained by a tracker.

Pixel-wise annotation [LSD15, HGD17, CPK17, WSY17, BM10] thrives over the recent years, as it provides a precise boundary of objects, especially irregularly shaped objects like landscape, sky, or sea, etc. Hence, many studies of dynamics at pixel level involves the estimation of optical flow [HS81, BA96, BBP04, ZPB07, BM11]. *Optical flow* is typically estimated between consecutive frames without partitioning the image domain into objects, but [WA94, SWS13] showed how to explicitly isolate regions corresponding to piece-wise parametric motion, whereas [YSS15, CMP07, BMT05, BS09] allow generic deformations. In particular, [YSS15, RM07, BWS09, CF13] require some form of initialization (manual or

automated pre-processing of the flow and images).

Video Segmentation problem partitions the video frames into non-overlapping regions [RM07, BT09, GKH10a, VAP10, XXC12, GCS12, ZJS13, PF13, OMB14, XYW11], while video object segmentation attempts to assign a single label to each object [OMB14, SSB15] or the foreground [LKG11, ZJS13, CCB06, PF13]. One added benefit of some of these methods is to produce, with the segmentation, depth layers [DP91, WA94, BM98, JF01, SDC04, JYS08, KTZ08, SSB12, SC12, CF13, YS13] that are informative of occlusion relations in the scene. Most of these methods, however, involve non-convex optimization. On the other hand, cascading motion estimation and depth ordering, while sub-optimal, can be framed as a sequence of two convex optimization problems and solved efficiently [CP11].

Dense 3-D reconstruction of surface geometry [KBC12, ND10, GPB11, KSC15, VSR15] is also related to this work, as a reconstructed scene trivially yields a partition of the video into regions. However, unlike our work, 3-D reconstruction approaches are typically restricted to static scenes and cannot handle multiple independently moving and deforming objects.

It is worth mentioning that while many approaches operate *offline* (or *non-causally*), with the entire video available for processing [OMB14, LKG11, ZJS13, PF13], our method processes the video causally and can scale to very long or even "endless" videos (as is the case with most autonomous agent applications). However, our method fails to detect objects that are not moving or attached to their background. This is by design, as we wish to distinguish, say, a car in the scene from a poster image of a car. Nevertheless, this behavior is penalized in benchmark evaluations. Our method also processes the data causally, since we also target closed-loop robotic applications, so it is naturally at a disadvantage compared to methods that process the entire video as a batch. Despite these challenges in video segmentation benchmarks, we achieve state-of-the-art performance.

5.2 Methodology

The key idea of this work is to leverage topological and regularity constraints imposed by the physical scene, namely occlusion relations, connectedness, and regularity of the induced domain deformations, to partition the video into "objects."

We employ a pseudo-measurement module that leverages local (in space and time) occlusion relations to provide an update (detector) on a running estimate of object shape maintained as the state of a tracker. The interplay between detector and tracker is critical: if one had a perfect detector, the tracker would extrapolate to the next frame; vice-versa, if one had the perfect tracker, it would be easy to detect objects. In practice, we leverage existing methods for detection and tracking, and integrate them in a manner that improves on their simple concatenation, allowing one to recover from failures of the other, as shown in Fig. 5.1. The result is a fully automatic pipeline that is competitive with the state-of-the-art video object segmentation benchmarks.

Specifically, at each temporal snapshot t, given measurements up to t as $\boldsymbol{Y}^t$ and (the point estimate of) the objects' state at time t as $\hat{\boldsymbol{x}}_t \doteq \{\boldsymbol{r}_t[i]; i = 0, \dots, N_t\}$ (Fig. 5.2, row 1), a *prediction* module (tracker) serves to predict the (foreground) objects' state at next time as $\tilde{\boldsymbol{x}}_{t+1} \doteq \{\tilde{\boldsymbol{x}}_{t+1}[i], i = 1, \dots, N_t\}$ (Fig. 5.2, row 2)[1]; a *pseudo-measurement* module (object detector) processes the next frame's observation $\boldsymbol{y}_{t+1}$ or small batch of frames (for us, 3) to produce object hypotheses or "proposals", $\{\boldsymbol{o}_t[j]; j = 1, \dots, M_{t+1}\}$ (Fig. 5.2, row 3); and an *update* module to generate updated estimate at next time frame, $\hat{\boldsymbol{x}}_{t+1} \doteq \{\hat{\boldsymbol{x}}_{t+1}[i]; i = 0, \dots, N_{t+1}\}$ (marked in orange in Fig. 5.2). The state update module should ensure that visibility constraints are satisfied, so each point on the image $\boldsymbol{y}_{t+1}$ corresponds to a single object, which is equivalent to enforcing opacity of objects in the scene. Initially, we assume that the scene is empty, i.e., $\boldsymbol{x}_0 \doteq \emptyset$.

[1]Although $\hat{\boldsymbol{x}}_t$ and $\tilde{\boldsymbol{x}}_t$ are both point estimates, we use $\hat{\boldsymbol{x}}_t$ to represent the estimate of the *posterior* state (i.e., state after updates) and $\tilde{\boldsymbol{x}}_t$ as *prior* state (i.e., predicted state from previous time frame without observation at current time).

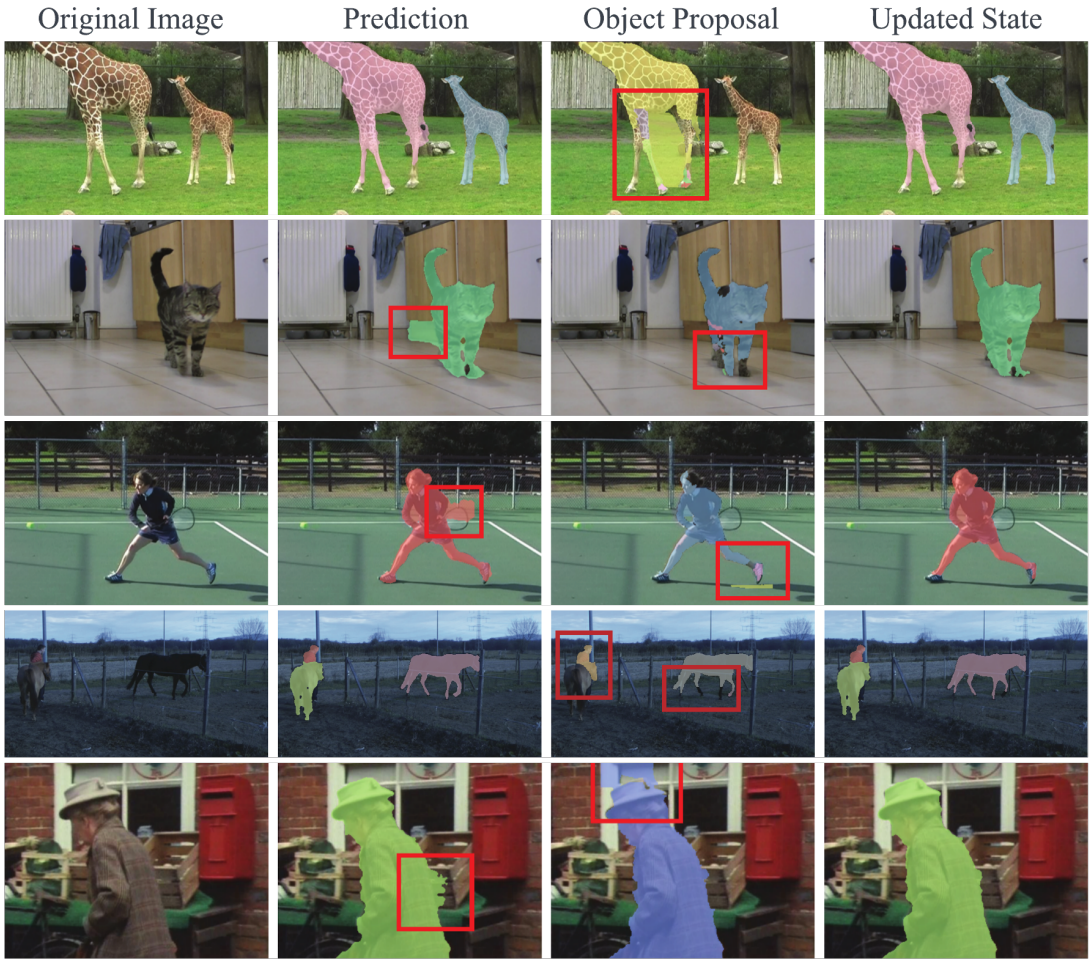

Figure 5.1: Illustration of the update of tracking and detection. Errors made by prediction or detection (marked in red rectangles) are corrected in the final estimate.

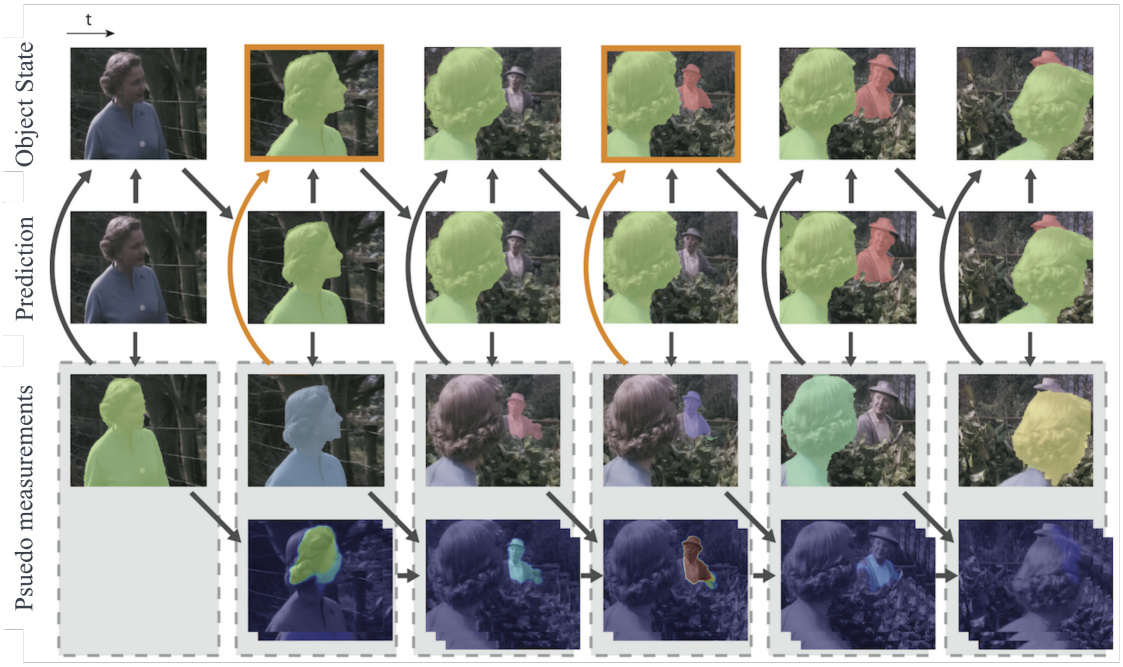

Figure 5.2: Illustration of our framework causally processing a video. The pseudo-measurement (object detector) (bottom 2 rows) proposes regions in the image that correspond to objects in the scene, which are integrated with the prediction (row 2) into the final object labeling (row 1). We begin with no objects in the state (left column) and an object is only added to the state after being detected across multiple frames (highlighted by orange arrows).

Note that we only process foreground objects in the prediction and pseudo-measurement modules, so the object index starts from 1. Also note that the number of objects at current time N_t, object hypotheses at next time M_{t+1}, and objects maintained in the next time's posterior state estimate N_{t+1}, can be different.

5.2.1 Prediction

Given the current state of objects $\{\hat{\boldsymbol{x}}_t[i]; i = 1, \dots, N_t\}$, we predict their next location $\{\tilde{\boldsymbol{x}}_{t+1}[i]; i = 1, \dots, N_t\}$ by warping the current regions with optical flow $\boldsymbol{f}_t$. The occlusion portion of each object region $\mathcal{O}_t[i]$ should be predicted and removed, while dis-occluded parts $\mathcal{D}_{t+1}[i]$ be estimated and added, yielding the operation as:

$$\tilde{\boldsymbol{x}}_{t+1}[i] \doteq \mathcal{W}\Big(\hat{\boldsymbol{x}}_t \setminus \mathcal{O}_t[i]\,;\, \boldsymbol{f}_t\Big) \bigcup \mathcal{D}_{t+1}[i]; \quad \forall i = 1, \dots, N_t \tag{5.1}$$

where $\mathcal{W}(\cdot; \boldsymbol{f})$ denotes warping operation with optical flow $\boldsymbol{f}$; $\setminus$ is the set minus; and $\bigcup$ the set union operator. The computation of inter-frame optical flow leverages on the assumptions of photometric persistence of objects and illumination: radiance changes slowly relative to the temporal sampling rate of the video. Additional assumptions such as boundary regularity and connectedness are needed to infer the occluded and dis-occluded regions. Since there exists extensive literature on the topic, rather than developing our own, we employ a recent deformable shape tracker that automatically infers dis-occlusions [YSS15], for which code is available online. It is important to note that this method assumes manual initialization, whereas in our case it is initialized using the current state of objects. Since each object is predicted independently, the update phase must manage conflicts among different objects predicted to overlap in the next image, in order to make our method viable.

5.2.2 Object detection

The pseudo-measurement module provides object proposals $\boldsymbol{O}_{t+1} \doteq \{\boldsymbol{o}_{t+1}[j]; j = 1, \dots, M_{t+1}\}$ using the image at time $t + 1$ or a small batch of images (short-term memory; long-term

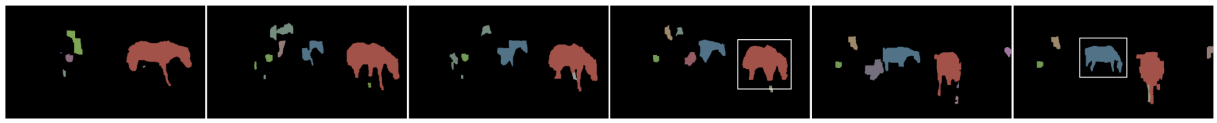

Figure 5.3: Object detections from our pseudo-measurement module. Object detection generates many false positives in this difficult scenario containing multiple deforming objects with complex motion dynamics. In this work, we trust consistently appearing regions to correspond to true objects in the scene. The accumulation of evidence from detections identifies the red segment (by the 31^{th} frame) and the blue segment (by the 48^{th} frame) as real objects and allows us to ignore the remaining proposals as noisy measurements. In this figure, the colors of consistent regions correspond across frames for ease of viewing, but in the actual system, correspondence is determined by the matching procedure in Sec. 5.2.2.

memory is represented by the object's state) to produce hypotheses of objects informed by occlusion relations. These exploit assumptions of the spatial connectedness of objects and the scene's topology. Object proposals also spawn new objects that appear and are not attributed to existing predicted ones. Occlusions have been used extensively to prime object detection or layer segmentation [WA94, BM98, AS12]. Again, since our approach is agnostic to the proposed method, we employ the last, for which code is available online. Note that, although [AS12] appears instantaneous, the method employs two or three temporally adjacent frames. In this case, the prediction is not fed back to the detector, but one could adopt different pseudo-measurement schemes where the prediction is used to guide the detection of the new image. Note that the prediction produces the inter-frame motion or deformation as a byproduct, including the forward motion as $\boldsymbol{f}_{t\to t+1}$ and the backward motion as $\boldsymbol{f}_{t+1\to t}$. if three frames are employed, thus closing the loop in our tracker.

To be robust to the gross failure of the pseudo-measurement module, we require objects to be detected in multiple frames before they are added to the objects' state in the tracker. This is useful when the motion or the scene is very complex (see Fig. 5.3 for an example

where multiple horses are moving independently), as occlusion detection and subsequently object detection from occlusions can be considerably noisy in these cases. This reduces the number of false positives in our output and helps keep the computational cost of prediction, which scales linearly with the number of objects tracked, manageable.

Thus, object proposals are accumulated over time before being declared as a true object. For the accumulation, each proposal $\boldsymbol{o}_t[j]$ is associated with a continuity confidence score as:

$$\mathrm{cf}_t\big(\boldsymbol{o}_t[j]\big) = \frac{\int_{\partial \boldsymbol{o}_t[j]} e(\boldsymbol{p}_t) d\boldsymbol{p}_t}{\mathcal{A}\big(\boldsymbol{o}_t[j]\big)} \in \mathbb{R}^1 \tag{5.2}$$

for all $j = 1, \ldots, M_t$, where $\mathcal{A}(\cdot)$ is the area of a region, $\boldsymbol{p}_t$ represents a pixel on the t^{th} video frame, $\partial \boldsymbol{o}_t[j]$ defines the boundary of object $\boldsymbol{o}_t[j]$, and $e(\boldsymbol{p}_t) \in \mathbb{R}^1$ measures the intensity and motion discontinuity strength, defined as

$$e(\boldsymbol{p}_t) \doteq \max\Big(c_1 e_{\text{img}}(\boldsymbol{p}_t) + c_2 e_{\text{motion}}(\boldsymbol{p}_t)\,,\, \varepsilon\Big) \tag{5.3}$$

where e_{img} and e_{motion} are the output of an edge detector on the image data and motion field, respectively, and $0 < \varepsilon \ll 1$ a constant number. We then associate each proposal $\boldsymbol{o}_{t+1}[j]$ from the current frame with a warped region $\tilde{\boldsymbol{o}}_{t+1}[k] \doteq \mathcal{W}\big(\boldsymbol{o}_t[k]; \boldsymbol{f}_t\big)$ from the past[2] if $\boldsymbol{o}_{t+1}[j]$ and $\tilde{\boldsymbol{o}}_{t+1}[k]$ achieves a sufficient overlap as measured by the Intersection-over-Union (IoU) score:

$$\mathrm{IoU}\big(\boldsymbol{o}_{t+1}[j], \tilde{\boldsymbol{o}}_{t+1}[k]\big) = \frac{\boldsymbol{o}_{t+1}[j] \cap \tilde{\boldsymbol{o}}_{t+1}[k]}{\boldsymbol{o}_{t+1}[j] \cup \tilde{\boldsymbol{o}}_{t+1}[k]} \tag{5.4}$$

Each matched proposal contributes its shape and score to the accumulated proposal. We define the accumulated measurement likelihood of the j^{th} object as a running score computed recursively as:

$$\begin{aligned} s_{t+1}\big(\boldsymbol{o}_{t+1}[j]\big) \doteq \sum_{\boldsymbol{p}_{t+1}} & \mathrm{cf}_{t+1}\big(\boldsymbol{o}_{t+1}[j]\big)\, \mathbb{1}\big[\boldsymbol{p}_{t+1} \in \boldsymbol{o}_{t+1}[j]\big] \\ & + s_t\big(\boldsymbol{o}_t[k_j]\big)\, \mathbb{1}\big[\boldsymbol{p}_{t+1} \in \tilde{\boldsymbol{o}}_{t+1}[k_j]\big] \end{aligned} \tag{5.5}$$

[2]Here we abuse the notation $\tilde{\boldsymbol{o}}_{t+1}[k]$ to represent the current-time object proposal $\boldsymbol{o}_t[k]$ warped by a forward optical flow $\boldsymbol{f}_t[k]$, which is computed during prediction. We use the index k to distinguish with j as the indices of warped objects from t and the detected objects of $t+1$ can have different order.

where $\mathrm{cf}_{t+1}(\cdot)$ is defined in Eqn. 5.2; k_j is the index of the object at time t with the best IoU score with $\boldsymbol{o}_{t+1}[j]$, i.e., $k_j \doteq \arg\max_{k=1}^{M_t} \mathrm{IoU}\big(\boldsymbol{o}_{t+1}[j], \tilde{\boldsymbol{o}}_{t+1}[k]\big)$, and $\tilde{\boldsymbol{o}}_{t+1}[k_j]$ is the region of warping the object $\boldsymbol{o}_t[k_j]$ to time $t+1$ with optical flow $\boldsymbol{f}_t$. The initial likelihood is set to the continuity confidence score, i.e., $s_{t_0}\big(\boldsymbol{o}_{t_0}[j]\big) \doteq \sum_{\boldsymbol{p}_{t_0}} \mathrm{cf}_{t_0}\big(\boldsymbol{o}_{t_0}[j]\big) \cdot \mathbb{1}\big[\boldsymbol{p}_{t_0} \in \boldsymbol{o}_{t_0}[j]\big]$ when the proposal was first added into the proposal pool at time t_0. Finally, pixels with likelihood $s_{t+1}(\boldsymbol{o}_{t+1}[j])$ exceeding a nominal threshold $\lambda = 0.5$ are promoted to objects added to the objects state maintained by the tracker and removed from the proposal pool.

5.2.3 State update

For each warped object $\tilde{\boldsymbol{x}}_{t+1}[i]$, we first determine its persistency from the past by associating it with any of the detected object proposals with $j_i \doteq \arg\max_{1 \le j \le M_{t+1}} \mathrm{IoU}\big(\boldsymbol{o}_{t+1}[j], \tilde{\boldsymbol{x}}_{t+1}[i]\big)$. The object proposals failed to match a presented one (new objects) are added into our updated state directly, while we established a conditional random field (CRF) model [LMP01] to generate the updated estimate $\hat{\boldsymbol{x}}_{t+1}[i]$ (Fig. 5.4.e) of persisted ones based on its predicted mask $\tilde{\boldsymbol{x}}_{t+1}[i]$ and detected mask $\boldsymbol{o}_{t+1}[j_i]$ (Fig. 5.4.d), yielding the updated state as a combined set:

$$\widehat{\boldsymbol{X}}_{t+1} \doteq \underbrace{\big\{\hat{\boldsymbol{x}}_{t+1}[i]; i = 0, \ldots, N_t\big\}}_{\text{persisted objects}} \bigcup \underbrace{\big\{\boldsymbol{o}_{t+1}[i]; i = N_t + 1, \ldots, N_{t+1}\big\}}_{\text{new objects}} \tag{5.6}$$

Fully connected conditional random field is introduced to refine the region of persist objects. The fusion of predicted and proposed object masks is formulated as a multi-class labeling function that maps the image domain $\boldsymbol{y}_{t+1} \in \mathbb{R}^{H \times W}$ to class labels $\boldsymbol{l}_{t+1} \doteq \big\{l_{t+1}[a]; 1 \le a \le H \times W\big\}$ given combined region $\boldsymbol{u}_{t+1}[i] \doteq \tilde{\boldsymbol{x}}_{t+1}[i] \cup \boldsymbol{o}_t[j_i]$, $1 \le i \le N_t$, where each element $l_{t+1}[a] \doteq l\big(\boldsymbol{p}_{t+1}[a]\big) \in \{0, 1, \ldots, N_t\}$ is the label assigned to the a^{th} pixel $\boldsymbol{p}_{t+1}[a]$. Note $l[\cdot] = 0$ refers to the label of background.

Objective function of solving the i^{th} object region is defined as:

$$\mathcal{J}_i(\boldsymbol{l}_{t+1}) = \sum_{a \le H \times W} \varphi_{\text{unary}}\big(l_{t+1}[a] = i\big) + \sum_{\substack{a<b \\ a,b \le H \times W}} \varphi_{\text{pair}}\big(l_{t+1}[a], l_{t+1}[b]\big) \tag{5.7}$$

where φ_{unary} is the unary potential and φ_{pair} is the pairwise potential.

Pairwise term incorporates the regularity of image edge topology and optical flow edge topology (Fig. 5.4.b) as:

$$\begin{aligned}\varphi_{\text{pair}}\Big(l_{t+1}[a], l_{t+1}[b]\Big) \doteq \mathbb{1}\Big[l_{t+1}[a] = l_{t+1}[b]\Big] \times \\ \left[\alpha_I \exp\Big(-\frac{\xi_I\big(\boldsymbol{p}_{t+1}[a], \boldsymbol{p}_{t+1}[b]\big)}{2\theta_I^2}\Big) + \alpha_F \exp\Big(-\frac{\xi_F\big(\boldsymbol{p}_{t+1}[a], \boldsymbol{p}_{t+1}[b]\big)}{2\theta_F^2}\Big)\right]\end{aligned} \tag{5.8}$$

where $\xi_I(\cdot)$ and $\xi_F(\cdot)$ represent the geodesic distance on image and optical flow edge graph, respectively; $\mathbb{1}\Big[l_{t+1}[a] = l_{t+1}[b]\Big]$ is the label compatibility function of $\boldsymbol{p}_{t+1}[a]$ and $\boldsymbol{p}_{t+1}[b]$.

Unary term is simply a weighted indicator function revealing if $\boldsymbol{p}_{t+1}[a]$ falls into the union region of detected and predicted mask of the i^{th} object:

$$\varphi_{\text{unary}}\Big(l_{t+1}[a] = i\Big) \doteq w_i[a] \cdot \mathbb{1}\Big[\boldsymbol{p}_{t+1}[a] \in \boldsymbol{u}_{t+1}[i]\Big] \tag{5.9}$$

where the weight $w_i[a] \in \mathbb{R}^1$ represents how we trust the results of (the union of) detection and prediction mask at $\boldsymbol{p}_{t+1}[a]$. Specifically, $w_i[a]$ should be bigger if $\boldsymbol{p}_{t+1}[a] \in \tilde{\boldsymbol{x}}_{t+1}[i] \cap \boldsymbol{o}_{t+1}[i]$ or $\boldsymbol{p}_{t+1}[a] \notin \tilde{\boldsymbol{x}}_{t+1}[i] \cup \boldsymbol{o}_{t+1}[i]$, and should be smaller if the predicted and detected labels disagree as $\boldsymbol{p}_{t+1}[a] \in \big(\tilde{\boldsymbol{x}}_{t+1}[i] \setminus \boldsymbol{o}_{t+1}[i]\big) \cup \big(\boldsymbol{o}_{t+1}[i] \setminus \tilde{\boldsymbol{x}}_{t+1}[i]\big)$.

Geodesic distance measures the distance of a graph's two vertices as the shortest (weighted) path between them. Ideally, solving Eqn. 5.7 evolves computing geodesic distance of arbitrary two pixels on a 2-D edge map, which is extremely costly. To solve this, we introduce a super-pixel clustering model to assign each pixel $\boldsymbol{p}_{t+1}[a]$ to the center of its corresponding super-pixel (Fig. 5.4.c). We abuse the notation $\boldsymbol{S}\boldsymbol{p}_{t+1}[a]$ as the super-pixel the pixel $\boldsymbol{p}_{t+1}[a]$ belongs to, whose centroid is computed as the average of pixel coordinates along with the spatial axis. Then approximate the geodesic distance as:

$$\begin{aligned}\xi_\iota\big(\boldsymbol{p}_{t+1}[i], \boldsymbol{p}_{t+1}[j]\big) = \xi_\iota\big(\boldsymbol{p}_{t+1}[i], \boldsymbol{p}^c_{t+1}[i]\big) + \xi'_\iota\big(\boldsymbol{p}^c_{t+1}[i], \boldsymbol{p}^c_{t+1}[j]\big) \\ + \xi_\iota\big(\boldsymbol{p}^c_{t+1}[j], \boldsymbol{p}_{t+1}[j]\big)\end{aligned} \tag{5.10}$$

where $\iota \in \{I, F\}$, $\xi'_\iota(\cdot)$ is a graph-based approximation of the geodesic distance of the centroid of two super-pixels, as described in [RWH15]. The distance $\xi_\iota\big(\boldsymbol{p}_{t+1}[i], \boldsymbol{p}^c_{t+1}[i]\big)$ is set to a

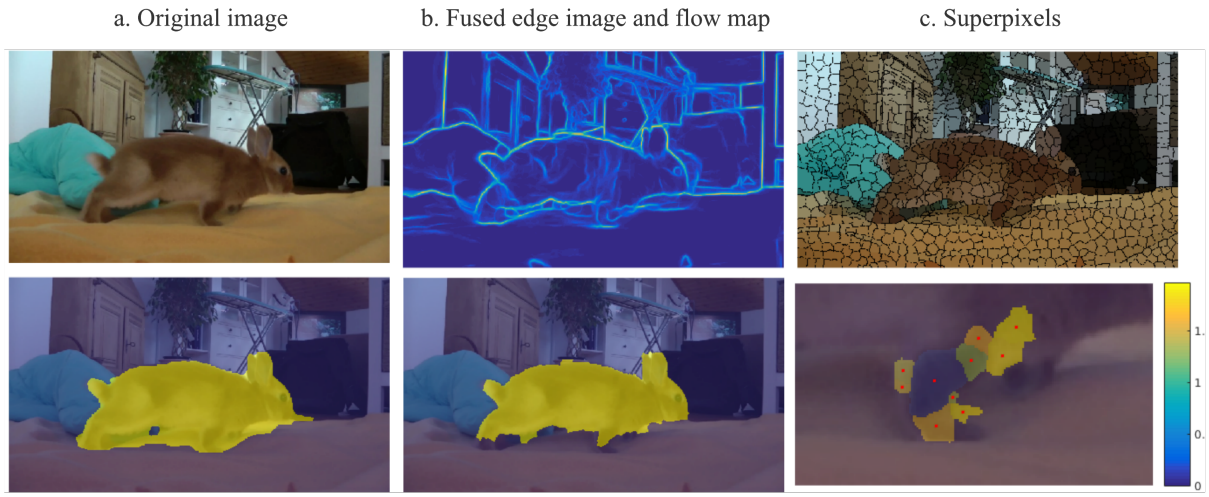

Figure 5.4: Illustration of updated estimate of object regions. (f.) demonstrates a super-pixel located on the rabbit leg (masked in blue) and its top 10 nearest super-pixel neighbors. The red dots represent the centroid of super-pixels respectively.

constant small value ϵ according to the smoothness property of a super-pixel, which states that the image boundary rarely cut across inner super-pixels.

We further assume that pixels belongs to the same super-pixel share the same label. Therefore, computing the unary term φ_{unary} and pairwise term φ_{pair} are of complexity $O\big(\bar{\mathcal{A}}(\boldsymbol{Sp}_t)\big)$ and $O\big(\bar{\mathcal{A}}(\boldsymbol{Sp}_t)^2\big)$, respectively, where $\bar{\mathcal{A}}(\boldsymbol{Sp}_t)$ is the average area of all the super-pixels of frame t. We further set the scalar $w_i[a]$ to infinity in Eqn. 5.9 if the super-pixel $\boldsymbol{Sp}_t[a]$ locates far away from the object boundary with a threshold, i.e., $w_i[a] \doteq +\infty$ if $\min_{\boldsymbol{p}_t[b]\in\boldsymbol{u}_t[i]} \|\boldsymbol{p}_t^c[a], \boldsymbol{p}_t[b]\|_2 > \gamma$. In practice, the number of super-pixels we use to compute is ≤ 350.

5.3 Experiments

5.3.1 Implementation detail

Our method segments video into object labels. We evaluate our approach on two popular datasets, MoSeg [OMB14] and BVSD [GNJ13]. We used the toolbox of [DZ13] for edge detection on both the image and motion field to obtain e_{image} and e_{motion} mentioned in Eqn. 5.3. The linear weights are set to $c_1 = 0.1, c_2 = 0.9$ for MoSeg and $c_1 = 0.3, c_2 = 0.7$ for BVSD. ε is set to 0.0001. θ_I and θ_F in Eqn. 5.8 are both set to 0.9, and the geodesic distance matrix is normalized by its mean.

Timing. Our method takes approximately 2 minutes for a VGA frame with 3 objects in the state on a single core and around 1 minute per frame when running in parallel on a standard desktop (8 core i7-3770 3.4GHz with 16GB RAM).

Detection. We modified the publically available code from [TKS15] to yield object pseudo-measurements, which takes approximately 30s per frame.

Tracking. We leveraged the publically available code of [YSS15] to perform the prediction step. Given the labeled object regions, the method takes approximately 44s to estimate motion and predict each region at time $t + 1$, increasing slightly with the number of objects provided. The code can also be run in parallel, reducing to 15-20s per frame depending on the number of cores available.

Update. The per-object update and the energy minimization problem can be solved in less than 5 seconds using any convex optimization solver (we used [GB08, GB14]).

5.3.2 Quantitative and qualitative results

For MoSeg, the task is multi-label object segmentation in the video. The dataset contains 59 sequences containing anywhere from 19 to 800 frames, a small subset (3–41) are annotated with pixel-wise ground truth. Given that our approach operates causally and the pseudo-

measurement relies on occlusions, and thus motion, we are unable to detect objects before they move (see Fig. 5.2 where two ladies are detected over time). To combat this, since our tracking module provides motion estimates between consecutive frames, we accumulate the motion estimate across frames until it is sufficiently large for our pseudo-measurement module to perform detection. This helps alleviate the lack of object detections near the beginning of videos in the datasets.

In order to avoid false detections from the pseudo-measurement module, our system only labels new objects when they have been detected multiple times. While valuable for ensuring reliable object hypotheses, this penalizes our system for evaluation, where the first frame is always annotated in the ground truth. To combat this issue, we back-propagate object labels captured by our system (as soon as they are added to the objects' state) to the first frame. Our tracking module can run this in a separate thread, which does not affect the casualty of our system.

The updated objects might overlap, which does not satisfy the general constraints of objects projected into the image plane. To ensure that only a single object label occupies each pixel, for each pixel in the overlapped area $\boldsymbol{p}_{t+1} \in \bigcap_{i=0}^{N_{t+1}} \boldsymbol{x}_{t+1}[i]$, we adopt the dominant label (i.e. the mode of labels) within top 10 neighbors based on geodesic distance.

Following the evaluation protocol of [OMB14], we report *precision*, *recall*, *F-measure*, and the number of extracted objects (labeled regions with F-measure ≥ 0.75) in Tab. 5.1. We outperform other state-of-the-art methods in recall and F-measure for both the training and testing splits of the dataset. In addition, our system discovers more objects, indicating the advantage of our system for building object hypotheses.

In Tab. 5.3, we show our quantitative results on the BVSD dataset. We report both boundary precision-recall (BPR), a commonly used metric in image segmentation, and volume precision-recall (VPR), which quantifies the spatio-temporal overlap of our labels and the ground truth regions. We obtain higher numbers than the other video object segmentation approaches. We do perform worse, however, than the video segmentation comparisons

	Training set (29 sequences)				Test set (30 sequences)			
	P	R	F	$N/65$	P	R	F	$N/69$
[GKH10b]	79.17	47.55	59.42	4	77.11	42.99	55.20	5
[OMB14]	81.50	63.23	71.21	16	74.91	60.14	66.72	20
[AS12]	87.20	59.60	70.81	17	79.64	50.73	61.98	7
[TKS15]	85.00	67.99	75.55	21	82.37	58.37	68.32	17
[TKS15]-NC	83.00	70.10	76.01	23	77.94	59.14	67.25	15
[YSS15]	**89.53**	70.74	79.03	26	**91.47**	64.75	75.82	27
ours	88.54	**76.54**	**82.11**	**29**	89.63	**69.69**	**78.41**	**32**

Table 5.1: Evaluation results on FBMS-59. Average precision (P), recall (R), and f-measure (F) over all sequences in the training and test datasets of FMS-59. Higher values indicate superior performance. All methods are fully automatic. Methods [TKS15] and our method are causal. The other methods process the whole video in batches.

	Training set (29 sequences)			Test set (30 sequences)		
	P	R	F	P	R	F
[AS12]	75.94	61.64	68.05	78.11	54.68	64.33
[LKG11]	64.86	52.70	58.15	62.32	55.97	58.97
[PF13]	71.34	70.66	71.00	76.29	63.29	69.18
[TKS15]	83.92	68.19	75.24	86.54	63.20	73.05
[TKS15]-NC	79.26	78.99	79.12	83.41	67.91	74.87
[YSS15]	**84.52**	75.47	77.11	86.13	74.08	77.68
ours	82.50	**81.56**	**79.95**	**86.68**	**81.26**	**82.27**

Table 5.2: Foreground-background segmentation results on MoSeg. Average precision (P), recall (R), and f-measure (F) over all sequences in the training and test sets from FBMS-59. [LKG11, PF13] and [TKS15]-NC process the video in batch, while [AS12, TKS15] and [YSS15] and our method are completely causal.

on this dataset, but this is expected as we cannot segment the background into multiple regions.

	Boundary			Volume		
	P	R	F	P	R	F
[OMB14]	0.566	0.100	0.170	0.146	0.852	0.249
[TKS15]	**0.760**	0.186	0.299	0.136	0.870	0.234
ours	0.697	**0.198**	**0.308**	**0.164**	**0.874**	**0.276**

Table 5.3: Quantitative results on BVSD dataset. Average precision (P), recall (R), and f-measure (F) over all sequences in the test datasets of BVSD. Higher values indicate superior performance.

5.4 Discussion of matching results with standard benchmarks

Although our goal is semi-supervised learning of objects, we do not evaluate our method end-to-end in object recognition benchmarks, since this entails a large number of engineering choices that would cloud the evaluation. Instead, we choose to evaluate our method as a video-object segmentation, using existing benchmarks, even though they penalize behaviors that are actually desirable in our actual goal. This includes penalizing the detection of objects that are present but not labeled in the dataset, and failure to detect objects that do not move. This is manifest in cases where objects move very little if at all in the benchmark videos, and therefore one cannot be sure if they are real or just poster images of them.

It should be noted that we wish to track the evolving occluding boundary of objects as a geometric entity, as that is informed by the shape of the object in 3-D. One could also track bounding boxes, and then delegate the encoding of geometric characteristics of objects (say shape) to photometric characteristics of the images (say the appearance of pixels within the bounding box), leaving the task of deciding which pixels within the bounding box belong to the actual object for later, in a multiple-instance learning framework. In our case, we make the separation of geometric and photometric properties explicit, which can be useful for the analysis and classification of objects that are defined solely by their shape, irrespective of reflectance, or vice-versa.

CHAPTER 6

Learning Semantic-Aware Dynamics

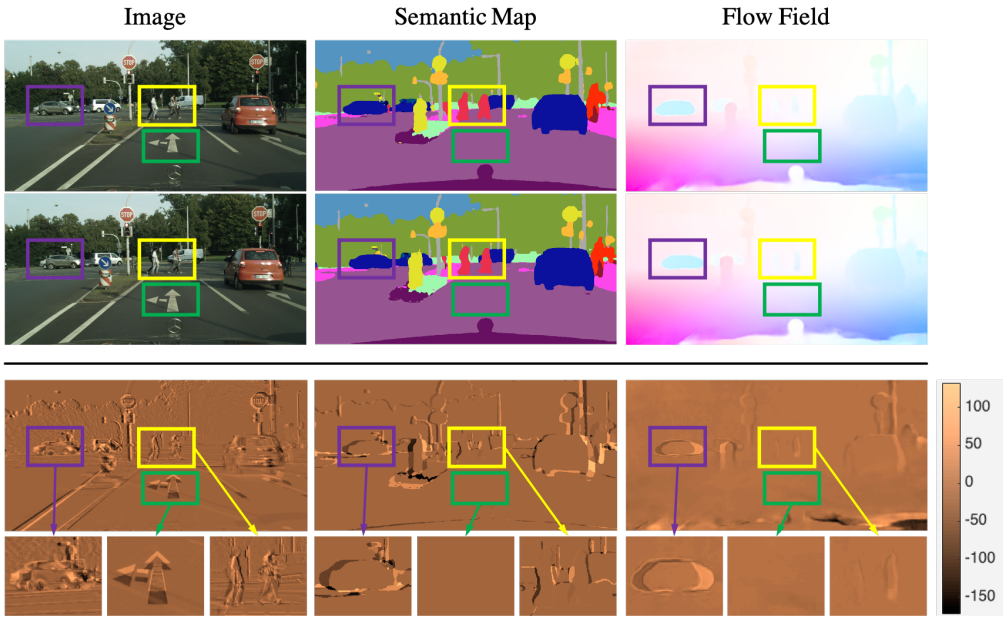

Figure 6.1: Different representations (video frame, semantic map, flow field) have dynamics with different complexity. **Top**: a sequence of video frames (left), semantic maps (middle), and flow fields (right). **Bottom**: dynamics or changes visualized in terms of their difference. The dynamics in video frames are much more complex than in semantic maps and flow fields.

So far, we modeled the scene dynamics from two different perspectives: Chapter. 4 learned task-specific representation that has no physical meaning, whose supervision is from the fully annotated data of the task variable; Chapter. 5 leverages generic regularities of the scene to decompose the video frames into object hypotheses, so there is no need (and no way) for the model to be trained with data.

The most informative function of past observations is the actual data itself. Instead of engineering the model architecture, we wish to capture generic regularities of video dynamics by learning a representation and enforcing it to predict future (observation) data, whose supervision is always available. This also fits into the self-supervised learning pipeline, whereas our self-supervised learning task and downstream task are the same - to predict the data itself. Hence, we validate the quality of the learned representation by achieving state-of-the-art performance on both video prediction (i.e., pixel intensity prediction) and semantic segmentation prediction tasks.

Given that the video data is high-dimensional, introducing assumptions or hypotheses of the environment is inevitable. We hypothesizes that decomposing the scene into independent entities is beneficial to prediction, each with its attributes. For example, in Fig. 6.1, different objects have different geometry and motion, which induces distinctive temporal changes in the video.

We propose a video prediction architecture that explicitly models the different dynamics of semantically consistent regions (Fig. 6.2). The model, described in Sec. 6.2.1, decomposes the video into regions, corresponding to different semantic classes in the scene, and learns class-specific characteristics while ensuring that their re-composition can predict the image, along with class labels and flow fields.

Unlike warping the past using globally predicted flow fields [LLD17, LFY18, PWJ19, GXC19], in our semantic-aware dynamic model (SADM), local regions are represented by binary semantic masks, whose evolution is simpler and easier to learn than the motion of the entire video frames (see Fig. 6.1). Each region is predicted and then fused with its content to generate future semantic maps and flow fields. The prediction in co-visible regions of future frames is warped from the past, with dis-occlusion detection mediated by the predicted semantic maps. Furthermore, the dis-occluded regions are filled in by a generative model or conditional renderer, trained with not only the warped images, but also the predicted semantic maps, enabling a more structured and semantically-aware synthesis. Modeling

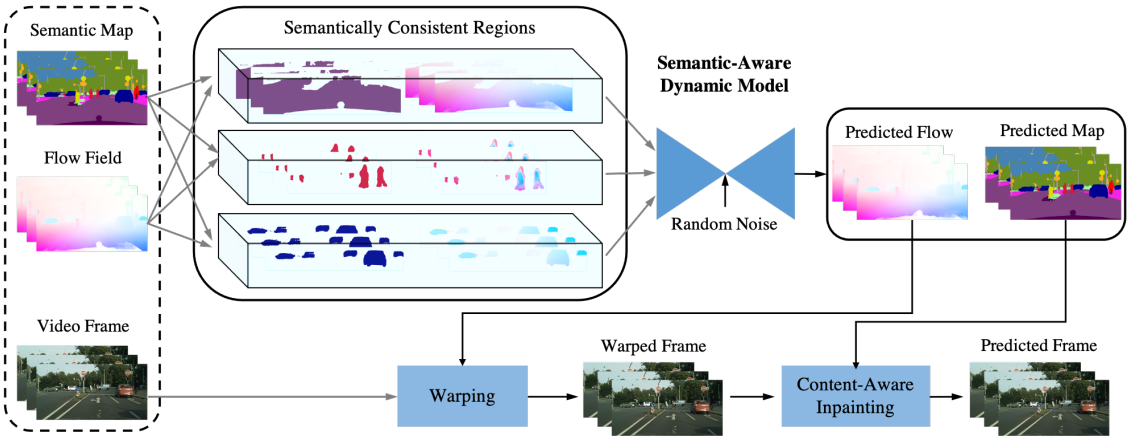

Figure 6.2: Our video prediction architecture with learned semantic-aware dynamics. It first decomposes the scene into semantically consistent regions to facilitate the modeling of class-specific characteristics. Using the proposed semantic-aware dynamic model, each region is predicted and fused to generate the future scene layout (semantic map) and motion (flow field). Content-aware video inpainting for dis-occlusions is performed after warping to generate the future video frames.

dis-occlusions explicitly spares the model the effort otherwise needed to learn this complex phenomenon.

6.1 Notion of task: video prediction

Video generation and prediction tasks aim to synthesize video frames' pixel values. Video generation creates videos from white noise or style preferences, while the video prediction task predicts future frames conditioned on past (observed) frames. We are more interested in the latter one, which is challenging from two perspectives: one must generate high-quality images at each temporal snapshot of the video while ensuring the video clips have consistent and realistic dynamic motion.

Notation. Given a clip of (observed) video frames $\boldsymbol{\mathcal{I}}^T \doteq \{\boldsymbol{I}_1, \ldots, \boldsymbol{I}_T\}$ up to time T, where each element $\boldsymbol{I}_t \in \mathbb{R}^{H \times W}$ is an image frame at time t, the task is to predict (not yet

observed) data up to future time $T+k$ with $k>1$, i.e., $\boldsymbol{\mathcal{I}}_{T+1}^{T+k} \doteq \{\boldsymbol{I}_{T+1}, \ldots, \boldsymbol{I}_{T+k}\}$. Note that, unlike the above chapters, we use T to denote the current time step (i.e., the last time step with observations available.), to distinguish with t that refers to any time frame within the video clip $\{1, \ldots T+k\}$.

Video generation methods produce image sequences either from noise [VPT16] or other input including pose [CBT18] and text [MMB17]. SVG-LP [DF18] proposes to sample noise from learned priors; MoCoGAN [TLY18] samples latent variables from the motion and content spaces separately to improve temporal consistency. Similarly, TGAN [SMS17] employs a temporal generator and an image generator to model temporal correlations; [HLM18] models the dynamics in the latent space with attribute controls. Given that the visual scene is highly structured, [VYZ17, CBT18, YWZ18] propose to generate a sequence of poses, which are transformed into images for human action sequences; [ZPT18] generates videos of a single object by first generating a sequence of conditions using a 3-D morphable model, while [HHB18] controls the video generation using sparse trajectories specified by the user. VGAN [VPT16] trains video generators with explicit separation of the foreground and background, assuming static background. Seg2vid [PWJ19] resorts to warping using flows generated by the semantic mask, hoping to preserve the scene structure implicitly. We also employ semantic maps to generate future flows but with a semantic-aware dynamic model. [YCS20] decomposes images into objects utilizing contextual information separation [YLS19] and synthesizes the motion of single objects through perturbations in the object-centric latent space.

Video prediction models are typically approximations of conditional generative models [HLH18, BFE17, WVE18, VPT16, SMS17, DF18, TLY18, RLS18, XNL18]. The quality of predictions is typically evaluated by image quality and temporal consistency. Given the high complexity and dimensionality of the signal to be predicted, the process usually requires explicit modeling or constraints [PWJ19, GXC19]. PredNet [LKC16] proposes a predictive model with coding-based regularization. ContextVP [BWK18] uses a context-aware module

with parallel LSTMs. SDC-Net [RLS18] applies flow guided spatially-displaced convolutions, while [JDT16] predicts with dynamic filters that depend on the inputs. DDPAE [HLH18] and [WVE18] map the observed images to a low-dimensional space, so temporal correlations are easier to learn. TPK [WMG17] predicts future poses to guide appearance changes. To address the loss of realism, [PGS14, LLD17, LPH17, LFY18] explicitly model the flows, and DVF [LYT17] uses flow to synthesize future frames. Similar to MCNet [VYH17] and [XWB16], DPG [GXC19] proposes motion-specific propagation and motion-agnostic generation with confidence-based occlusion maps. [LNC17] predicts future semantic maps, and [JXS17] jointly predicts the future semantic maps and flow fields. We use a semantic-aware model such that the predicted maps can exploit class-specific motion priors. Our method generates future optical flows and semantic maps before rendering future images. In [WGP20], moving object segmentation masks are used but restricted to 2-D affine motions, with two categories: moving and static.

6.2 Methodology

In the modeling, we incorporate semantic segmentation (scene layout), optical flow (scene motion) and synthesis (scene appearance) into a complete generative model for videos, which facilitates semantically and geometrically consistent prediction of complete video frames. Specifically, besides the directly observed video frames $\boldsymbol{\mathcal{I}}^T$, we aim to explicitly model the semantic aware dynamics of both the semantic maps $\boldsymbol{M}^T \doteq \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T\}$ and flow fields $\boldsymbol{F}^T \doteq \{\boldsymbol{f}_1, \ldots, \boldsymbol{f}_T\}$, where each element $\boldsymbol{m}_t \in [0, C]^{H \times W}$ and $\boldsymbol{f}_t \in \mathbb{R}^{H \times W \times 2}$ are the semantic mask and flow field measure of the video frame $\boldsymbol{I}_t \in \mathbb{R}^{H \times W}$, $1 \leq t \leq T$. And C is the number of semantic classes in the semantic map.

Thus, the problem can be reformulated as: given past measurements $\boldsymbol{Y}^T \doteq \{\boldsymbol{I}^T, \boldsymbol{M}^T, \boldsymbol{F}^T\}$ up to time T, our task variable $\boldsymbol{z}_t$ at time t is to predict k frames into the future, i.e., $\boldsymbol{z}_t \doteq \boldsymbol{Y}_{T+1}^{T+k} \doteq \{\boldsymbol{\mathcal{I}}_{T+1}^{T+k}, \boldsymbol{M}_{T+1}^{T+k}, \boldsymbol{F}_{T+1}^{T+k}\}$. The predictions should match the statistics, quality,

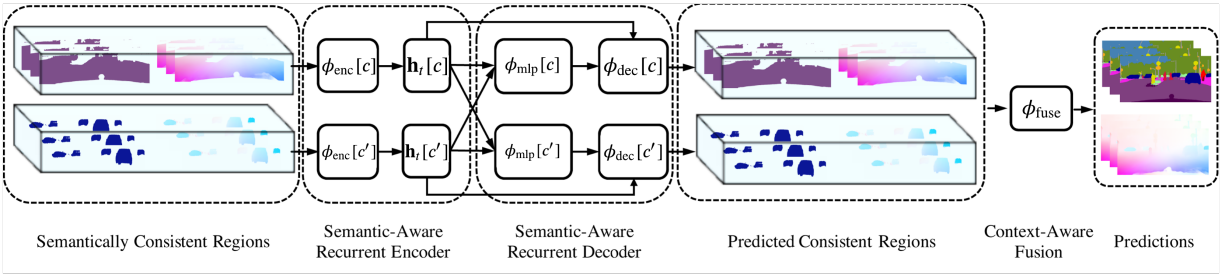

Figure 6.3: The architecture of our semantic-aware dynamic model (SADM) for learning class-specific dynamics of the scene layout and motion. Input semantic maps and flow fields are parsed and processed by the semantic-aware recurrent encoders ϕ_{enc} and decoders ϕ_{dec}, with context incorporated into the prediction through a multi-layer perceptron. The predictions of semantically consistent regions are combined by the fusion network ϕ_{fuse} to generate the prediction on the whole image domain, which further improves contextual compatibility in the predicted semantic maps and flow fields. The illustration is for two classes, but can be easily extended to more classes.

and content of past frames of the same scene and exhibit variations consistent with the motion of objects within. As illustrated in Fig. 6.2, our approach falls in the direction of prediction by propagation, where video prediction for the co-visible part[1] of the scene can be accomplished by warping, i.e., propagating pixels via the corresponding flow field.

6.2.1 Semantic-aware dynamic model

In general, the proposed semantic aware dynamic model takes as the flow fields and semantic maps up to time T, i.e., $\{\boldsymbol{M}^T, \boldsymbol{F}^T\}$ and outputs the future k (estimated) flow fields and semantic maps $\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}, \widehat{\boldsymbol{F}}_{T+1}^{T+k}\}$, as shown in Fig. 6.3. The components therein are elaborated on below.

Recurrent encoder. Let $\boldsymbol{m}_t[c] = \mathbb{1}[\boldsymbol{m}_t = c] \in [0,1]^{H\times W}$ be the binary mask that

[1]Image regions that are observed/visible across multiple frames

indicates the region of semantic class c at time t. Similarly, $\boldsymbol{f}_t[c] = \boldsymbol{f}_t \circ \boldsymbol{m}_t[c] \in \mathbb{R}^{H\times W\times 2}$ is the masked flow field[2] showing only the motion of the pixels that are classified as c. The semantic aware recurrent encoder $\phi_{\text{enc}}[c]$ operates recursively to produce a hidden representation from the past $\left\{\boldsymbol{M}^T[c], \boldsymbol{F}^T[c]\right\}$ while enforcing temporal continuity of the representation[3]:

$$\boldsymbol{h}_{t+1}[c] = \phi_{\text{enc}}[c]\Big(\big[\boldsymbol{m}_{t+1}[c], \boldsymbol{f}_{t+1}[c]\big], \boldsymbol{h}_t[c]\Big) \tag{6.1}$$

for all $1 \leq t \leq T$, with $\boldsymbol{h}_t[c]$ refers to the hidden representation of an RNN that summarizes the past regions and flow fields of the pixels within class c, up to time t. In other words, $\boldsymbol{h}_t \doteq \left\{\widehat{\boldsymbol{x}_{\text{m},t}}, \widehat{\boldsymbol{x}}_{\text{f},t}\right\}$ is an (point) estimate of the hidden state $\boldsymbol{X}^T$ in terms of mask and flow field measurements as in our generic pipeline in Sec. 3.2.

For now, we instantiate C such semantic aware recurrent encoders $\phi_{\text{enc}}[c]$ with $0 \leq c \leq C$, which together generate the hidden representation $\boldsymbol{h}_t \doteq \left\{\boldsymbol{h}_t[c]; c = 0, \ldots, C\right\}$ that summarizes the past semantic maps and flow fields, covering all semantic classes. Note that the collection $\boldsymbol{h}_t$ explicitly represents independent objects. While this may appear inefficient, in reality, the model reduces the number of parameters needed since the individual objects are simpler to represent. We also carry out an ablation study in Sec. 6.3.3 on different C's by merging some of the semantically similar classes, showing the accuracy-efficiency trade-offs. Moreover, we can efficiently parallelize the computation using the grouped convolution operator proposed in [KSH17]. Next, we describe the procedure to predict the future semantic maps and flow fields.

Recurrent decoder. Given the hidden representation of the past, $\boldsymbol{h}_t$, the semantic aware recurrent decoder produces k estimated future semantic maps and flow fields $\left\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}, \widehat{\boldsymbol{F}}_{T+1}^{T+k}\right\}$. We first describe a deterministic decoding procedure, for simplicity, which can then be easily adapted to a stochastic one to account for the randomness of the future.

[2]The operator $\circ$ is the Hadamard product (or elemen-wise product) of two matrices.

[3]Here we abuse the notation of $\phi_{\text{enc}}[c]$ to refer to the c^{th} component of function group ϕ_{enc} that applies to the c^{th} semantic category. Similarly, $\boldsymbol{M}^T[c]$ and $\boldsymbol{F}^T[c]$ represents the set of masked semantic masks and flow fields of the c^{th} category, at time from 1 to T, respectively.

Again, we consider decoders that learn the dynamics and predict the future in a semantic aware manner. Let $\phi_{\text{dec}}[c]$ be the recurrent decoder for semantic class c, which generates the estimates for $\{\boldsymbol{M}_{T+1}^{T+k}, \boldsymbol{F}_{T+1}^{T+k}\}$ by recursively executing the following procedures:

$$\boldsymbol{h}_{t+1}[c], \boldsymbol{e}_{t+1}[c] = \phi_{\text{dec}}[c]\Big(\boldsymbol{h}_t[c], \phi_{\text{mlp}}[c]\big(\boldsymbol{h}_t\big)\Big); \tag{6.2}$$

$$\widehat{\boldsymbol{m}}_{t+1}[c] = \phi_{\text{dec,m}}[c]\big(\boldsymbol{e}_{t+1}[c]\big); \quad \widehat{\boldsymbol{f}}_{t+1}[c] = \phi_{\text{dec,f}}[c]\big(\boldsymbol{e}_{t+1}[c]\big) \tag{6.3}$$

for all $T+1 \leq t \leq T+k$. Here we abuse the notation $\phi_{\text{dec}}[c]$ to refer to the recurrent unit that updates the latent representation $\boldsymbol{h}_t[c]$, while generating a common embedding $\boldsymbol{e}_t[c]$, which is then decoded into the predicted semantic mask $\widehat{\boldsymbol{m}}_t[c]$ and flow fields $\widehat{\boldsymbol{f}}_t[c]$, respectively through separate decoding heads $\phi_{\text{dec,m}}[c]$ and $\phi_{\text{dec,f}}[c]$. This separate decoding design aligns with the practice that improves the decoding efficiency in multi-task learning. Note, we also apply a multi-layer perceptron $\phi_{\text{mlp}}[c]$ (due to its efficiency) on the collection of the hidden representations for all classes $\big\{\boldsymbol{h}_t[c]; c = 0, \ldots, C\big\}$, to ensure that the semantic aware decoder has access to the context provided by other classes within the scene (Fig. 6.3).

The decoders for each class $\big\{\phi_{\text{dec,m}}[c], \phi_{\text{dec,f}}[c]; c = 0, \ldots, C\big\}$ can also be running in parallel, so that we have the semantic aware predictions for each class, i.e., $\big\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}[c], \widehat{\boldsymbol{F}}_{T+1}^{T+k}[c]\big\}$ with $c = \{0, \ldots, C\}$. Next, we apply late fusion to get predictions that can be directly compared to the ground truth semantic maps and flow fields, and to further improve the contextual compatibility between different classes.

Context-aware late fusion. Given the predicted $\big\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}, \widehat{\boldsymbol{F}}_{T+1}^{T+k}\big\}$ for each $c \in \{1, ..., C\}$, we apply a three-layer ConvNet ϕ_{fuse} to first fuse the binary semantic maps:

$$\widehat{\boldsymbol{m}}_t = \phi_{\text{fuse}}\Big(\big[\widehat{\boldsymbol{m}}_t[1], \ldots, \widehat{\boldsymbol{m}}_t[C]\big]\Big) \tag{6.4}$$

for all $T+1 \leq t \leq T+k$, where the dimension of $\widehat{\boldsymbol{m}}_t$ is $H \times W \times C$, $[\cdot, \cdot]$ is the operation of concatenation. We use softmax as the last layer for ϕ_{fuse}, such that each slice of $\widehat{\boldsymbol{m}}_t$ indexed by the last dimension, is still a scalar field indicating the probability of each pixel belonging

to class c. And the fused flow field $\widehat{\boldsymbol{f}}_t$ is obtained as following:

$$\widehat{\boldsymbol{f}}_t = \sum_{c=0}^{C} \mathbb{1}\left[\widehat{\boldsymbol{m}}_t = c\right] \circ \widehat{\boldsymbol{f}}_t[c] \tag{6.5}$$

which is a linear combination of the flow vectors predicted by each semantic aware recurrent decoder, whose visibility comes from the fused semantic map.

Training loss. With the ground truth semantic mask and optical flow at $t+1$ up to $t+k$, the training loss for flow fields is the $L1$ loss:

$$\mathcal{L}_{\text{flow}} = \frac{1}{k} \sum_{t=T+1}^{T+k} \|\widehat{\boldsymbol{f}}_t - \boldsymbol{f}_t\|_1 \tag{6.6}$$

which penalizes the discrepancy between the predicted flow and the ones computed from the ground truth images. For the semantic maps we apply the cross entropy loss:

$$\mathcal{L}_{\text{mask}} = \sum_{t=T+1}^{T+k} (1 + \alpha\ \mathcal{G} * \nabla \boldsymbol{m}_t) \cdot \mathcal{X}(\boldsymbol{m}_t, \widehat{\boldsymbol{m}}_t) \tag{6.7}$$

Here $\mathcal{X}$ represents the cross-entropy, which is weighted by whether the pixel is near the boundaries between different classes or not. Note that ∇ is the gradient operator, and we binarize its response to 0 and 1 to discount the artifacts caused by naming different classes with different integers. The binarized boundary map is then smoothed by a Gaussian kernel $\mathcal{G}$ to expand the weights to nearby pixels, making the boundaries thicker. The variance of the Gaussian, which determines the spatial extent of the boundaries, is set to 9.0 and fixed. With this weighting scheme, the network will focus more on the pixels near the semantic boundaries, thus better preserving the shape of each semantic segment in the prediction. The relative importance between boundary and non-boundary pixels is controlled by the scalar α, which is set to 5.0 for all experiments.

The stochastic model. So far, we have described the proposed semantic aware dynamic model in its deterministic mode. However, extending it to account for the stochasticity of the future is straightforward. For this purpose, we instantiate C semantic aware recurrent

encoders $\phi'_{\text{enc}}[c]$, which operate in a similar way as the encoders for the past:

$$\boldsymbol{r}_{t+1}[c] = \phi'_{\text{enc}}[c]\Big(\boldsymbol{m}_{t+1}[c], \boldsymbol{f}_{t+1}[c], \boldsymbol{r}_t[c]\Big) \tag{6.8}$$

at training time, or

$$\boldsymbol{r}_{t+1}[c] = \Big[\boldsymbol{\mu}_{t+1}[c], \boldsymbol{\Sigma}_{t+1}[c]\Big] \tag{6.9}$$

at inference time. The goal of the recurrent encoder $\phi'_{\text{enc}}[c]$ is to generate a random variable $\boldsymbol{r}_t[c]$, represented by its mean $\boldsymbol{\mu}_t[c]$ and variance $\boldsymbol{\Sigma}_t[c]$ through reparameterization, whose initial value is set to $\boldsymbol{r}_t[c] \doteq \big[\widehat{\boldsymbol{h}}_t[c], \mathbf{1}\big]$.[4] At the end of the recursion, we would like $\big\{\boldsymbol{r}_{t+1}[c], \ldots, \boldsymbol{r}_{t+k}[c]\big\}$ to be a zero-mean unit-variance Gaussian. At each recursion step $t > T$, $\big\{\big[\boldsymbol{\mu}_t[c], \boldsymbol{\Sigma}_t[c]\big]; c = 0, \ldots, C\big\}$ will be added to $\boldsymbol{h}_t$ in Eqn. 6.2, also through reparameterization, for decoding the future with randomness. To learn $\phi'_{\text{enc}}[c]$'s, we add a KL-divergence term to the loss:

$$\mathcal{L}_{\text{kl}} = \sum_{t=T+1}^{T+k} \mathbb{KL}\Big(\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \mathcal{N}(\mathbf{0}, \mathbf{1})\Big) \tag{6.10}$$

where $\mathcal{N}$ represents the normal distribution. We summarize the training loss for the stochastic semantic aware dynamic model in the following:

$$\mathcal{L}_{\text{dynamic}} = \mathcal{L}_{\text{flow}} + \mathcal{L}_{\text{mask}} + \beta\mathcal{L}_{\text{kl}} \tag{6.11}$$

with β the weight on the KL-divergence term. As in VAEs, $\big\{\phi'_{\text{enc}}[c]; c = 0, \ldots, C\big\}$ are used only during the training for the stochastic decoder, and will not be used during testing since the random noise can be directly sampled from the prior $\mathcal{N}(\mathbf{0}, \mathbf{1})$.

6.2.2 Warping with semantic informed dis-occlusion

We warp the past video frames to provide an anchor point for future synthesis using the predicted future semantic masks and flows fields $\big\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}, \widehat{\boldsymbol{F}}_{T+1}^{T+k}\big\}$ from the semantic-aware dynamic model. To ease the warping and comply with the literature, here we mark the

[4]This ensures that the generation of the random variable is conditioned on the past.

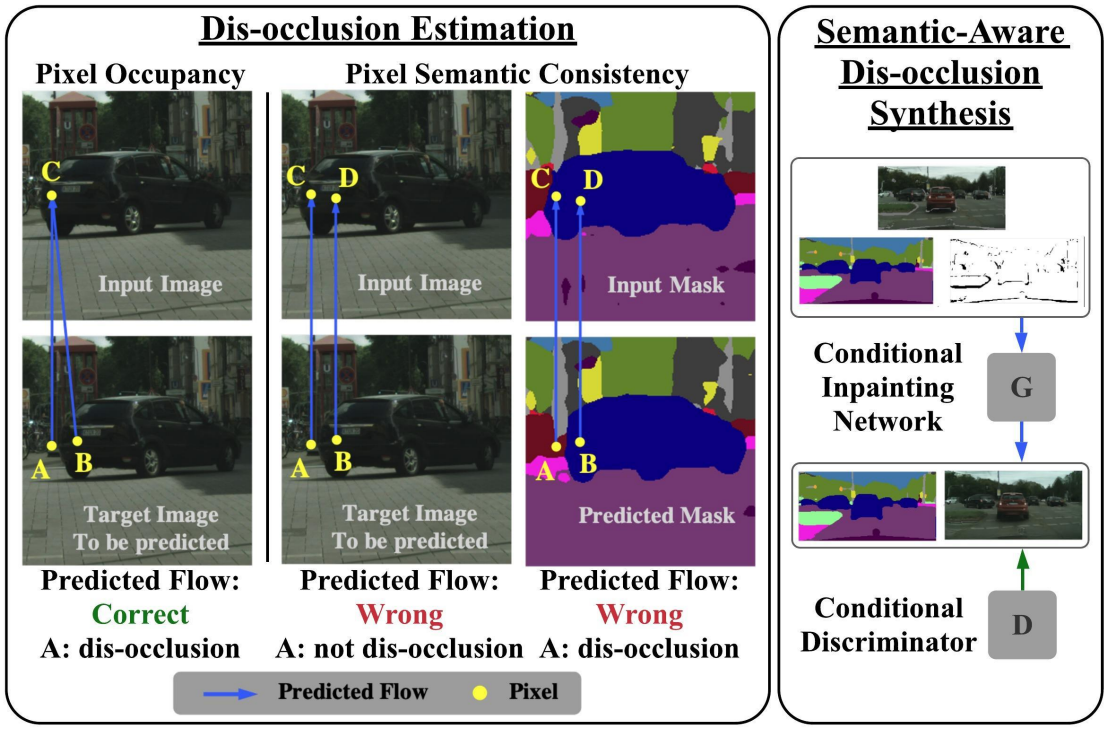

Figure 6.4: **Left**: Two criteria for dis-occlusion detection. *Pixel Occupancy*: pixels A, B in the target image domain are mapped onto pixel C in the input image domain, which is occupied by more than one pixel when the predicted (backward) flow (blue arrows) is correct; in this case, pixel A, as the cause of over-occupancy, can be detected as dis-occlusion. *Pixel Semantic Consistency*: if the predicted flow is incorrect, pixel occupancy fails in detecting A as dis-occlusion; however, given that semantic mask can be accurately predicted, A will be mapped to pixel C with inconsistent semantic labels, thus can be correctly detected as dis-occlusion. **Right**: Semantic-aware dis-occlusion synthesis, where both the generator and the discriminator take in the predicted semantic mask, and the generator is dis-occlusion aware.

predicted flow $\widehat{\boldsymbol{f}}_t$ as the backward flow, i.e., $\widehat{\boldsymbol{f}}_{t+1\to t}$. The warping can be simply performed via bilinear interpolation[5]:

$$\tilde{\boldsymbol{I}}_{t+1}(\boldsymbol{p}_{t+1}) = \boldsymbol{I}_t\Big(\boldsymbol{p}_{t+1} + \widehat{\boldsymbol{f}}_t(\boldsymbol{p}_{t+1})\Big), \quad \forall \boldsymbol{p}_{t+1} \notin \mathcal{D}_{t+1} \tag{6.12}$$

where $\boldsymbol{I}_t\big(\boldsymbol{p}_t\big)$ and $\boldsymbol{f}_t\big(\boldsymbol{p}_t\big)$ represents the pixel value and optical flow value at pixel $\boldsymbol{p}_t$. The key is to estimate the dis-occluded area $\mathcal{D}_{t+1}$, which invalidates the assumption that a pixel in frame $\boldsymbol{I}_{t+1}$ is propagated from the previous frame $\boldsymbol{I}_t$.

Note, [GXC19] proposes to use pixel occupancy for dis-occlusion detection, however, miss-detection happens due to errors in flow prediction on the object boundaries where dis-occlusion resides (see Fig. 6.4). Given that semantic masks are easier to predict than flows, particularly, with our semantic-aware dynamic model, we propose a semantic consistency criteria for dis-occlusion estimation, i.e., $\boldsymbol{p}_{t+1} \in \mathcal{D}_{t+1},$ if $\widehat{\boldsymbol{m}}_{t+1}(\tilde{\boldsymbol{p}}_{t+1}) \neq \boldsymbol{m}_t(\boldsymbol{p}_t)$. Here $\tilde{\boldsymbol{p}}_{t+1} = \mathcal{W}\big(\boldsymbol{p}_t; \boldsymbol{f}_t\big)$ is the pixel $\boldsymbol{p}_t$ warped by flow $\boldsymbol{f}_t$. The above semantic consistency criteria can still correctly detect dis-occlusions even if the flow is wrong, as shown in Fig. 6.4. Our experiments use both the pixel occupancy and the proposed semantic consistency criterion for dis-occlusion detection, given their complementarity.

After warping, we end up with the future frames warped from the past and the corresponding dis-occlusion masks, i.e., $\big\{\tilde{\boldsymbol{I}}_T, \mathcal{D}_{T+1}, \ldots, \mathcal{D}_{T+k}\big\}$. Note $\tilde{\boldsymbol{I}}_t$ is only valid (up to noise) in the complement of $\mathcal{D}_t$, which will be extrapolated as we describe next.

6.2.3 Semantic-aware dis-occlusion synthesis

Using the warped frames $\widetilde{\boldsymbol{\mathcal{I}}}_{T+1}^{T+k} \doteq \big\{\tilde{\boldsymbol{I}}_{T+1}, \ldots, \tilde{\boldsymbol{I}}_{T+k}\big\}$ as the anchor, we employ a conditional inpainting network to further complete the dis-occluded parts and improve the quality of the synthesized images via adversarial training. The conditional inpainting network ϕ_{syn} takes as input the anchor frame $\tilde{\boldsymbol{I}}_t$, and tries to complete the missing region indicated by $\mathcal{D}_t$ based

[5]We use $\tilde{\boldsymbol{I}}_t$ to represent the warped image at time t and $\widehat{\boldsymbol{I}}_t$ as the final (synthesized) image after filling the dis-occlusion area.

on the predicted semantic map $\widehat{\boldsymbol{m}}_t$ in a content-aware manner:

$$\widehat{\boldsymbol{I}}_t = \phi_{\text{syn}}(\tilde{\boldsymbol{I}}_t, \mathcal{D}_t, \widehat{\boldsymbol{m}}_t) \tag{6.13}$$

for all $T+1 \leq t \leq T+k$. Note, image details and their temporal consistency can be improved by semantic maps informing the scene content as shown in [WLZ18], which only focuses on translating known semantic maps to images. Given the ability to model the dynamics of the semantic map and its prediction, our conditional inpainting network can be informed about the scene content, thus able to generate better synthesis (see Fig. 6.5).

To help the training of the content-aware conditional inpainting, we also employ two discriminators $\text{Dis}_{\text{video}}$, Dis_{img} for the video clip and frame respectively, with $\text{Dis}_{\text{video}}$ focuses on the temporal continuity and Dis_{img} focuses on the image quality. So the training loss for the content-aware inpainting network is:

$$\begin{aligned}\mathcal{L}_{\text{syn}} = \sum_{t=T+1}^{T+k} \big(\mathbf{1} - \mathcal{D}_t\big) \cdot \|\widehat{\boldsymbol{I}}_t - \tilde{\boldsymbol{I}}_t\|_1 + \lambda \mathcal{L}_{\text{per}}\big(\widehat{\boldsymbol{I}}_t, \boldsymbol{I}_t\big) \\ + \eta \text{Dis}_{\text{video}}\big(\widehat{\boldsymbol{\mathcal{I}}}_{T+1}^{T+k}, \widehat{\boldsymbol{M}}_{T+1}^{T+k}\big) + \gamma \sum_{t=T+1}^{T+k} \text{Dis}_{\text{img}}\big(\widehat{\boldsymbol{I}}_t, \widehat{\boldsymbol{m}}_t\big)\end{aligned} \tag{6.14}$$

where the first term measures the discrepancy between the completed image and the warped image in the co-visible area; the loss $\mathcal{L}_{\text{per}}$ measures the perceptual similarity between the generated images and the real images [PWJ19]. Dis_{img} and $\text{Dis}_{\text{video}}$ measure the plausibility of images/videos conditioned on the semantic content. The final predictions of our model are $\big\{\widehat{\boldsymbol{M}}_{T+1}^{T+k}, \widehat{\boldsymbol{F}}_{T+1}^{T+k}, \widehat{\boldsymbol{\mathcal{I}}}_{T+1}^{T+k}\big\}$, i.e., predicted semantic maps, flow fields and video frames with the semantic-aware dynamics model as their driving force.

6.3 Experiments

Datasets. We evaluate our method on multiple *prediction* tasks, e.g., video frames and semantic maps, using three commonly used datasets, Cityscapes [COR16], KITTI Flow [GLU12] and KITTI Raw [GLS13]. Cityscapes [COR16] contains driving sequences recorded

in 50 different cities. We use the training split for training our semantic-aware dynamic model and the validation set for evaluation. The training and evaluation subsets contain 2975 and 500 videos, respectively. Pixel-wise annotations for semantic segmentation are only available every 20 frames for the Cityscapes dataset. KITTI Raw [GLS13] contains 156 long sequences. Following [WGP20], we use 4 of them for testing and the rest for training. KITTI Flow [GLU12] is designed for benchmarking optical flow algorithms and is more challenging than KITTI Raw [GLS13]. It consists of 200 training videos and 200 test videos. Following [GXC19], we downsample the videos to 128×424 and then center-crop to 128×256, yielding 4000 clips for both training and testing. Since per-frame dense semantic maps and optical flow annotations are not available, we leverage the off-the-shelf semantic segmentation network DeepLabV3 [CZP18] to extrapolate annotations for 20 classes and compute the optical flow using the PWC-Net [SYL18].

Implementation and training. We adapt the grouped Conv-LSTM network [XCW15] for the semantic-aware recurrent encoders and decoders to perform temporal aggregation of semantic maps and flow fields. The inpainting network is a modified U-Net [RFB15], conditioned on predicted anchor frames and semantic maps. We also replace the convolutional layers in the encoder with the partial convolution proposed in [LRS18], which masks the dis-occluded area in the feature space at different resolutions. The video and image discriminators are similar to those in CycleGAN [ZPI17], except that the video discriminators have ordinary 2-D convolutions replaced with 3-D convolutions.

Though our model can be trained end-to-end, we split the training into two stages to manage on a single work-station: 1) Training the semantic-aware dynamic model with loss Eqn. 6.11; 2) Training the inpainting network to fill in the dis-occluded area with loss Eqn. 6.14. The training is performed on 2 GeForce GTX 1080 Ti GPUs with a batch size equals to 6, and each video clip in the batch contains 10 frames. Learning rates for both stages start from 0.001 and decay by 0.8 every 20 epochs. The training of the semantic aware dynamic model needs 40 hours to converge, and the training of the inpainting network takes about

Method	**MS-SSIM** (×1e-2) ↑		**LPIPS** (×1e-2) ↓	
	T+1	T+5	T+1	T+5
PredNet [LKC16]	84.03	75.21	25.99	36.03
MCNET [VYH17]	89.69	70.58	18.88	37.34
Voxel Flow [LYT17]	83.85	71.11	17.37	28.79
Vid2vid [WLZ18]	88.16	75.13	10.58	20.14
Seg2vid [PWJ19]	88.32	61.63	9.69	25.99
FVS [WGP20]	89.10	75.68	8.50	16.50
SADM	**95.99**	**83.51**	**7.67**	**14.93**

Table 6.1: Quantitative comparison on the Cityscapes dataset.

20 hours.

Model complexity and inference. The semantic-aware dynamic model for predicting semantic maps and flow fields contains 8.6M parameters. The inpainting network contains 5.18M parameters. Inference can be performed on a single GeForce GTX 1080 GPU with 8GB memory. The inference runs at 19 frames per second.

6.3.1 Quantitative results

Video prediction. Tab. 6.1 and Tab. 6.2 report the multi-frame video prediction performance evaluated in terms of Multi-scale Structural Similarity Index Measure (MS-SSIM) [WSB03] and LPIPS [ZIE18], on the Cityscapes and the KITTI Raw datasets respectively. Higher MS-SSIM scores and lower LPIPS distances suggest better performance. Specifically, on longer horizon prediction ($T + 5$), our model improves Seg2vid [PWJ19], which also employs semantic segmentation, by 35.50% (MS-SSIM) and 20.85% (LPIPS). Moreover, our model outperforms FVS [WGP20], which infers 2-D affine transformations of moving objects, by 10.35% (MS-SSIM) and 9.39% (LPIPS) on the $T + 5$ predictions.

Method	**MS-SSIM** (×1e-2) ↑			**LPIPS** (×1e-2) ↓		
	T+1	T+3	T+5	T+1	T+3	T+5
PredNet [LKC16]	56.26	51.47	47.56	55.35	58.66	62.95
MCNet [VYH17]	75.35	63.52	55.48	24.05	31.71	37.39
Voxel Flow [LYT17]	53.93	46.99	42.62	32.47	37.43	41.59
FVS [WGP20]	79.28	67.65	60.77	18.48	24.61	**30.49**
SADM	**83.06**	**72.44**	**64.72**	**14.41**	**24.58**	31.16

Table 6.2: Quantitative comparison on the KITTI Raw dataset.

Following DPG [GXC19], we report the next-frame prediction results on the KITTI Flow dataset in Tab. 6.3. For a fair comparison, we also include the commonly used Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [WBS04] as the evaluation metrics, in addition to Learned Perceptual Image Patch Similarity (LPIPS) [ZIE18]. Our model improves DPG [GXC19], the previous state-of-the-art method evaluated on this dataset, by 9.7% in terms of PSNR, 2.2% in terms of SSIM, and 4.4% in terms of LPIPS.

Semantic segmentation mask prediction. We evaluate our model's performance on semantic mask prediction using the Cityscapes dataset, with the standard mean Intersection-over-Union score (mIoU) as the evaluation metric. Following [LNC17], the scores are computed with respect to the ground truth segmentation of the 20^{th} frame in each sequence. Tab. 6.4 shows the semantic mask prediction performance on multiple prediction lengths. Our method performs the best among the other methods, especially when the prediction horizon gets longer.

6.3.2 Qualitative results

In Fig. 6.5, we compare to FVS [WGP20] and Seg2vid [PWJ19], two most recent methods that employ semantic segmentation or moving object segmentation to facilitate video pre-

Method	**PSNR**↑	**SSIM** (×1e-2)↑	**LPIPS** (×1e-2)↓
Repeat [GXC19]	16.5	48.9	19.0
PredNet [LKC16]	17.0	52.7	26.3
SVP-LP [DF18]	18.5	56.4	20.2
MCNet [VYH17]	18.9	58.7	23.7
MoCoGAN [TLY18]	19.2	57.2	18.6
DVF [LYT17]	22.1	68.3	16.3
CtrlGen [HHB18]	21.8	67.8	17.9
DPG [GXC19]	*22.3*	*69.6*	*11.4*
SADM	**24.47**	**71.1**	**10.9**

Table 6.3: Quantitative comparison in next-frame prediction on the KITTI Flow dataset.

diction. The motivation of Seg2vid [PWJ19] is that the high-level semantics of the scene will result in more accurate predictions. However, without explicit modelings, such as SADM, predicted flow fields from [PWJ19] still suffer from over-smoothing. Moreover, given that most of the videos in Cityscapes are captured by a camera moving forward with a car, there is a strong tendency in the model from Seg2vid to produce flow fields showing zooming-in motion. On the other hand, FVS [WGP20] divides the whole scene into moving and non-moving segments by moving object detection. Although 2-D affine transformations are predicted per frame to approximate the object motion, complex motion, including deformation and 3-D rotation, may not be captured by a single 2-D affine transformation. Even for non-moving rigid objects whose motion is induced by the camera motion, e.g., parked cars and buildings, their projected 2-D flow fields still depend on the 3-D geometries and thus are not 2-D affine. Our model can generate high-quality video frames with more accurate motions with semantic-aware dynamics and inpainting.

Method	T+1	T+5	T+9
Repeat	67.1	52.1	38.3
S2S-dil [LNC17]	-	59.4	47.8
PSPNet [NRW18]	71.3	60	-
Jin [JXS17]	66.1	-	-
Terwilliger [TBL19]	73.2	67.1	51.5
Bayes-WD-SL [BFS18]	**74.1**	65.1	51.2
F2MF-DN121 [al20]	-	69.6	57.9
SADM	73.8	**70.3**	**60.1**

Table 6.4: Quantitative results of semantic map prediction on the Cityscapes dataset measured by the mIoU score.

6.3.3 Ablation study

Effectiveness of semantic-aware layers. To demonstrate the effectiveness of decomposing the video into semantically consistent regions for video prediction, we train a baseline model ("single class" in Fig. 6.7) with the same network architecture as SADM, and with a naive concatenation of semantic masks and flow fields as the input to the baseline model. The encoder and decoder of this baseline model share similar structures as those in SADM, besides that ordinary convolutions are used. Note that an ordinary convolution layer has more parameters than a grouped convolutional layer ($O(K^2)$ v.s. $O(K)$). As shown in Fig. 6.7, without explicitly modeling the class-wise dynamics, the baseline model trained to predict flow fields or video frames has difficulties estimating the motion near the boundaries between different semantic regions (the black car in the top left). There is heavy over-smoothing near the boundary of the car, which is problematic for the consecutive warping procedure, since the flow there will warp pixels on the car to the background or vice versa, generating the "ghost effect". With the proposed semantic aware dynamic model, the motion of either the car or the background can be accurately estimated since the influence on motion estimation

from occlusions is automatically handled through the decomposition. Similarly, in the bottom left of Fig. 6.7, the warped image using the flow predicted by the baseline model shows far more artifacts on the red car.

To show that the learned dynamics from our model are indeed semantic-aware, we test the model with semantic labels intentionally swapped in the input. For example, we input the "car" segments to the semantic-aware encoder that learns the "road" class dynamics and vice versa. As expected (Fig. 6.6), the predicted motion of the "car" segments using the encoder for the "road" class (middle row) now looks like the one of the "road" (bottom row). Similarly, the "road" segments' predicted motion using the encoder for the "car" class (middle row) now looks more like the one from the "car" (bottom row). This shows that the proposed model captures exactly that semantic-aware dynamics.

Number of semantic classes. We have experimented with different numbers of classes (C) for training our semantic-aware dynamic model (SADM). Our observations are: 1) the performance of our model, measured in terms of the video prediction quality, is robust with respect to the number of classes C; 2) the optimal performance may not be achieved by the model which has access to the full range of semantic classes; The results we reported in the main paper are from our model trained with nine classes ($C = 9$), i.e., the 20 classes within the Cityscapes dataset include 12 static categories and 8 moving categories, and we merge the 12 static categories into a single class, resulting in 9 classes. In Tab. 6.5, we show the quantitative results of SADM trained with $C = 9$ and $C = 20$ (no merging of any classes), together with comparisons to other state-of-the-art methods on the Cityscapes dataset. One can observe that both of them achieve comparable or better performance than the other state of the arts. And SADM with $C = 9$ performs slightly better than SADM with $C = 20$.

Our conjecture is that the estimation error in the semantic maps generated by Deeplab or any off-the-shelf semantic segmentation networks have a larger impact on SADM with $C = 20$. For example, the segmentation boundary may not be accurate between two static classes and the errors could propagate. However, when these two static classes are merged,

Method	**MS-SSIM** (×1e-2) ↑		**LPIPS** (×1e-2) ↓	
	T+1	T+5	T+1	T+5
PredNet [LKC16]	84.03	75.21	25.99	36.03
MCNET [VYH17]	89.69	70.58	18.88	37.34
Voxel Flow [LYT17]	83.85	71.11	17.37	28.79
Vid2vid [WLZ18]	88.16	75.13	10.58	20.14
Seg2vid [PWJ19]	88.32	61.63	9.69	25.99
FVS [WGP20]	89.10	75.68	8.50	16.50
SADM-20	93.25	79.14	10.06	17.55
SADM-9	**95.99**	**83.51**	**7.67**	**14.93**

Table 6.5: Ablation study on the number of classes used for our semantic-aware dynamic model with comparisons to other state-of-the-art methods on the Cityscapes dataset.

the in-between erroneous boundaries just disappear, leaving no errors for propagation, which indeed can facilitate the training.

6.3.4 Computation complexity

As mentioned in Sec. 6.3, SADM use grouped convolution in the encoder/decoder and employ class-specific recurrent units. This breaks combinatorial complexity, can run in parallel and leaves the modeling of relations among classes to a separate fusion layer besides the context-aware MLP in the decoders. As discussed in Sec. 6.3.3, jointly modeling the dynamics of all classes would require much larger capacity and a much larger training set.

To provide a quantitative measurement of the complexity of our model, we compare the number of parameters w.r.t to several state-of-art models as in Tab. 6.6, from which we can make the following observations: 1) even with more parameters, SADM with single class ($C = 1$) still can not learn the dynamics well; 2) the number of parameters in SADMs with

$C = 9$ and $C = 20$ are comparable to the other state-of-the-art methods, which demonstrates that the learning efficiency of SADM does not come from any increase in the network capacity but purely from the explicit modeling of semantic-aware dynamics.

Method	**Num of parameters**
PredNet [LKC16]	6.9M
ContextVP [BWK18]	8.6M
DVF [LYT17]	8.9M
STMFA [JHT20]	7.6M
Seg2vid [PWJ19]	202M
SADM-9	8.6M
SADM-20	23.2M
SADM-ablation	38.1M

Table 6.6: Number of parameters in comparison with state-of-the-art models.

6.3.5 Failure cases

We have tested the hypothesis that representing object-level motion in a video can be beneficial for prediction. To that end, we have proposed a model that captures occlusions explicitly and represents class-specific motion. While such high-level modeling is beneficial to prediction, there are failure cases. Specifically, hallucinating the dis-occluded regions can lead to failure when the background is complex (Fig. 6.8). As the time horizon grows, the prediction becomes increasingly unrealistic, as with other video prediction models, but the explicit modeling of objects and class-specific motion yields improvements over generic models. Also, we are constrained by classes for which we have training data, which limits generalization. So, our work is only a first step to incorporating dynamic models informed by the semantics of objects in the scene, which we expect will ultimately facilitate intelligent interaction with physical scenes by autonomous agents.

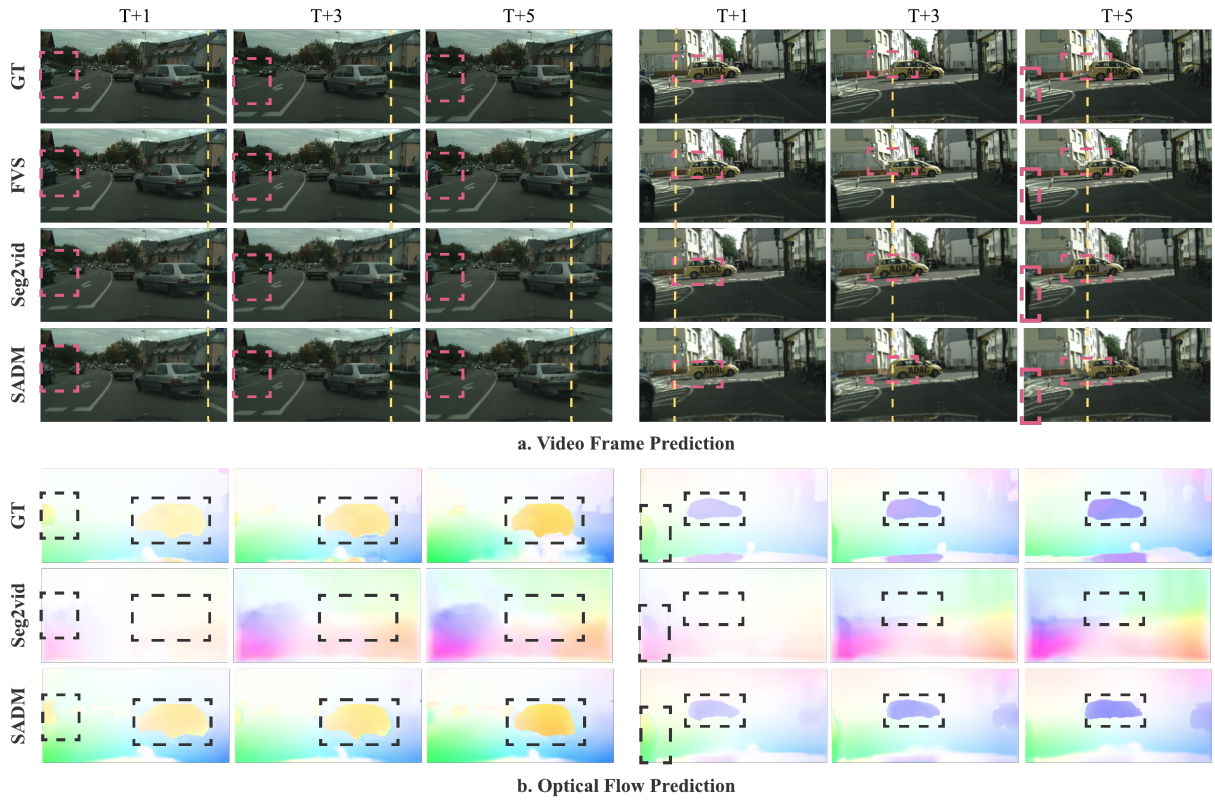

Figure 6.5: Visual comparison of the Cityscapes dataset. Both the predicted video frames (a) and flow fields (b) are presented. **Left**: the flow predicted by our network clearly shows the silver car moves to the left and the camera moves forward, while the flow predicted from Seg2vid [PWJ19] is dominated by "major" camera motion exhibited in the dataset, i.e., zooming-in caused by the movement of the running car. FVS [WGP20] wrongly predicts the motion of the silver car, resulting in incorrect car locations at $T + 3$ and $T + 5$. Note that, at $T + 5$, the black car on the left should move outside the image domain, which is only captured by our model. Again, the "ghost effect" presents near the objects' boundaries in the predictions from the other two methods. **Right**: without conditioning on semantic segmentation masks, the dis-occluded area of the building is incorrectly inpainted by the inpainting network as part of the moving car, causing distortions in the results from FVS.

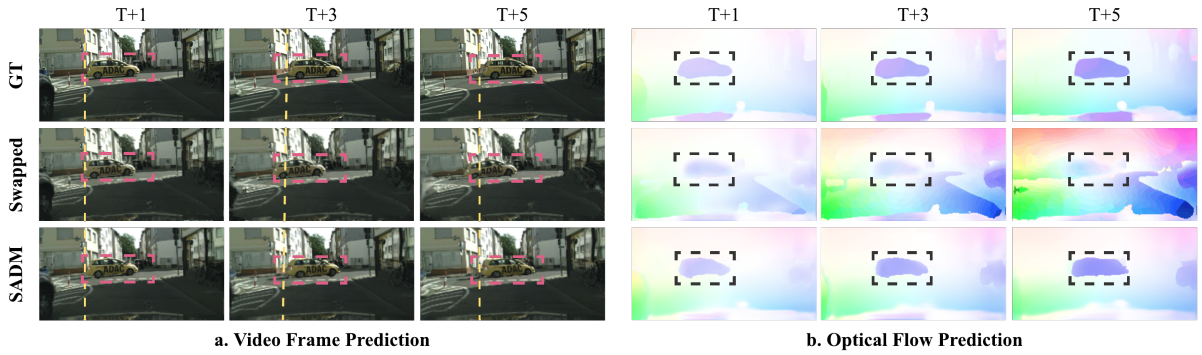

Figure 6.6: Ablation study: our model predicts video frames and flow fields by swapping semantic-aware encoders for the classes of "car" and "road", verifying that semantic-aware dynamics is learned with the proposed model.

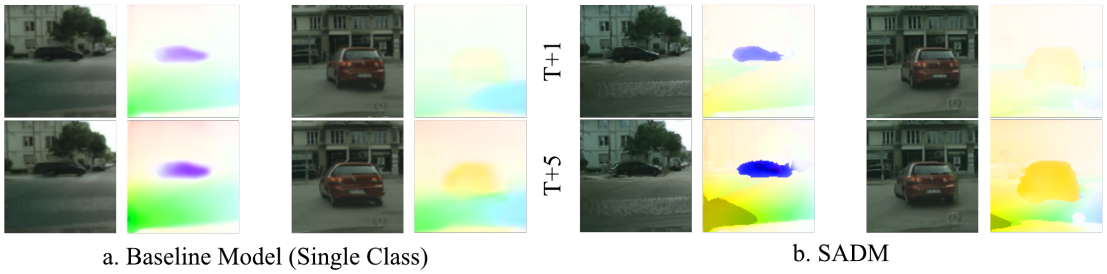

Figure 6.7: A baseline model without explicit modeling of the semantic-aware dynamics (single class) shows less accurate motion prediction than SADM and has more artifacts in the predicted video frames.

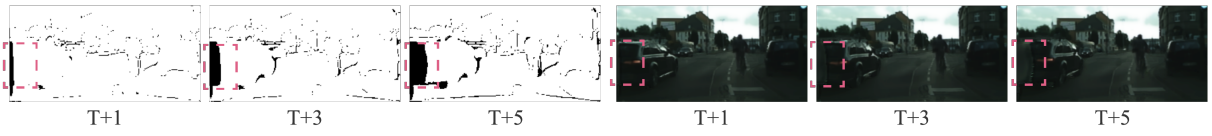

Figure 6.8: Complicated dis-occluded regions cause difficulties for the inpainting network, even if the estimated occlusions are accurate (left).

CHAPTER 7

Conclusion

We interpret human perception of the physical scene as building an estimate of past observations, updating the knowledge with current observations, and utilizing the learned knowledge to anticipate the future. Although the estimate, or "knowledge", is theoretically infinite-dimensional, we intend to approximate it with a finite-dimensional task-specific representation, that contains all the past information we need to predict the task in the future.

In the first part of our work, we prove that learning task-specific representation is tractable with a model realized in its most general form without heuristic assumptions on the physical meaning of the hidden state. We adopt neural networks as universally-approximating function classes to imitate the functionality of a state transition model and a measurement model, so that the hidden feature tensor maintained in the recurrent structure naturally becomes a task-specific representation of the scene.

Secondly, we demonstrate that the pipeline is still tractable even without any supervision from the task variable. The model was built upon generic regularities like temporal consistency or occlusion relations, to partition the scene into object regions and maintain a "meaningful" representation at the object level. Although showing good performance on standard benchmarks, the method requires complicated hand-crafted designs to transfer human inductive bias into executable machine modules.

Thirdly, we propose a model that benefits from both human inductive bias and prior knowledge learned from the vast amount of (un-annotated) raw video data. By enforcing class-specific pixel generators to predict future video data, we capture generic motion priors

in a semantic-aware manner.

Learning the representation is a trade-off between complexity and fidelity of the approximation of the "true state". The more abstract knowledge we capture from the data, the less data volume we need to carry around, the less fine-grained details we can preserve, and vice versa. Again, the extension of trading-off highly depends on the computation resources we have and the task at hand. Our work of learning task-sufficient representation from the video dynamics contributes new insights to this game of tug-of-war.

CHAPTER 8

Glossary of Notation

$\doteq$	Denote. For example, $\emptyset \doteq \{\}$ means we use notation $\emptyset$ to represent the concept "an empty set".
$\mathcal{P}$	The probability distribution.
t	The timestamp.
$\boldsymbol{x}_t$	A random variable at times t.
$\boldsymbol{X}^t$	A collection of random variables $\boldsymbol{x}$ from time 1 up to time t.
$\boldsymbol{X}_t^{t+k}$	A collection of random variables $\boldsymbol{x}$ from time t up to time $t+k$.
$\boldsymbol{x}[i]$	i^{th} element of a random variable $\boldsymbol{x}$.
$\boldsymbol{x}$	The latent state of a stochastic process.
$\boldsymbol{y}$	The observation variable of a stochastic process. In most cases, observations are video frames, $y \doteq I$.
$\boldsymbol{z}$	The task variable.
$\boldsymbol{I}_t$	Video frame at time t.
$\boldsymbol{\mathcal{I}}^t$	A collection of video frames up to time t.
$\boldsymbol{p}_t$	A pixel located on the t^{th} video frame.
$\boldsymbol{w}_{\text{foo}}$	The parameters of a module named as "foo".
$\boldsymbol{r}_t$	An image region (collection of adjacent pixels) at time t.
$\boldsymbol{Sp}$	A superpixel.
$\mathcal{A}(\cdot)$	The area of an image region.
$\boldsymbol{f}_t$	$\boldsymbol{f}_t \in \mathbb{R}^{H \times W \times 2}$, the optical flow defined on video frame t with size $H \times W$.

$\boldsymbol{m}_t$	$\boldsymbol{m}_t \in [0, C]^{H \times W}$, the semantic segmentation mask flow defined on the video frame t with size $H \times W$. C is the number of classes.
$\mathcal{O}$	The occluded region of an image. $\mathcal{O} \in [0, 1]^{H \times W}$ defined on the image plane with shape $H \times W$.
$\mathcal{D}$	The dis-occluded region of an image. $\mathcal{D} \in [0, 1]^{H \times W}$ defined on the image plane with shape $H \times W$.
$\setminus$	The removing operation by excluding all pixels belongs to region $\boldsymbol{r}[j]$ from $\boldsymbol{r}[i]$, i.e., $\boldsymbol{r}[i] \setminus \boldsymbol{r}[j] \doteq \boldsymbol{p} \in \boldsymbol{r}[i]$ and $\boldsymbol{p} \notin \boldsymbol{r}[j]$.
$\delta_a(\cdot)$	The Dirac measure at a given position a.
$\mathcal{N}(\boldsymbol{\mu}; \boldsymbol{\Sigma})$	A normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.
$\mathcal{W}(\cdot; \boldsymbol{f})$	The warping operation. e.g., $\mathcal{W}(\boldsymbol{I}; \boldsymbol{f})$ refers to warping the image I by optical flow values $\boldsymbol{f}$.
IoU	The intersection-over-union operator.
$\mathbb{1}[\cdot]$	The indicator function.
$\boldsymbol{\sigma}(\cdot)$	The sigmoid function. $\boldsymbol{\sigma}(x) \doteq 1/(1 + e^{-x})$
$\tanh(\cdot)$	The hyperbolic tangent function, i.e., $\tanh(x) \doteq (e^x - e^{-x})/(e^x + e^{-x})$
$*$	The convolution operator.
$\circ$	The Hadamard product (element-wise product).
$\boldsymbol{h}_t$	The hidden state of a recurrent neural network at time t.
$\mathcal{L}_{\text{foo}}$	A loss function term with name "foo".
$\mathbb{KL}$	The Kullback–Leibler divergence
$[\boldsymbol{a}, \boldsymbol{b}]$	The concatenation of (discretized) tensor $\boldsymbol{a}$ and $\boldsymbol{b}$.

[al20] Josip S. et al. "Warp to the Future." In *CVPR*, 2020.

[AS12] Alper Ayvaci and Stefano Soatto. "Detachable Object Detection: Segmentation and Depth Ordering From Short-Baseline Video." *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012.

[AS18] Alessandro Achille and Stefano Soatto. "A separation principle for control in the age of deep learning." *Annual Review of Control, Robotics, and Autonomous Systems*, **1**:287–307, 2018.

[BA96] M.J. Black and P. Anandan. "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields." **63**(1):75–104, 1996.

[BBP04] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. "High accuracy optical flow estimation based on a theory for warping." In *Eur. Conf. Comput. Vis.*, pp. 25–36. Springer, 2004.

[BFE17] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. "Stochastic variational video prediction." *arXiv preprint arXiv:1710.11252*, 2017.

[BFS18] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. "Bayesian prediction of future street scenes using synthetic likelihoods." *arXiv preprint arXiv:1810.00746*, 2018.

[BL95] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. YBs London, UK:, 1995.

[BM98] Lothar Bergen and Fernand Meyer. "Motion Segmentation and Depth Ordering Based on Morphological Segmentation." In *Eur. Conf. Comput. Vis.*, 1998.

[BM10] Thomas Brox and Jitendra Malik. "Object segmentation by long term analysis of point trajectories." In *European conference on computer vision*, pp. 282–295. Springer, 2010.

[BM11] Thomas Brox and Jitendra Malik. "Large displacement optical flow: descriptor matching in variational motion estimation." *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[BMT05] M.F. Beg, M.I. Miller, A. Trouvé, and L. Younes. "Computing large deformation metric mappings via geodesic flows of diffeomorphisms." *Int. J. Comput. Vis.*, **61**(2):139–157, 2005.

[BRL09] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, and Luc Van Gool. "Robust tracking-by-detection using a detector confidence particle filter." In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1515–1522. IEEE, 2009.

[BS09] Leah Bar and Guillermo Sapiro. "Generalized Newton-type methods for energy formulations in image processing." *SIAM Journal on Imaging Sciences*, **2**(2):508–531, 2009.

[BT09] William Brendel and Sinisa Todorovic. "Video Object Segmentation by Tracking Regions." In *Int. Conf. Comput. Vis.*, 2009.

[BVH16] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. "Fully-convolutional siamese networks for object tracking." In *European Conference on Computer Vision*, pp. 850–865. Springer, 2016.

[BWK18] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. "Contextvp: Fully context-aware video prediction." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 753–769, 2018.

[BWS09] Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. "Video snapcut: robust video object cutout using localized classifiers." In *ACM Transactions on Graphics (TOG)*, 2009.

[BYS21] Xinzhu Bei, Yanchao Yang, and Stefano Soatto. "Learning semantic-aware dynamics for video prediction." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 902–912, 2021.

[CBT18] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. "Deep video generation, prediction and completion of human action sequences." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 366–382, 2018.

[CCB06] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. "Bilayer Segmentation of Live Video." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2006.

[CCY] J Choi, HJ Chang, S Yun, T Fischer, Y Demiris, and JY Choi. "Attentional correlation filter network for adaptive visual tracking.".

[CF13] J. Chang and J. W. Fisher III. "Topology-Constrained Layered Tracking with Latent Flow." In *Int. Conf. Comput. Vis.*, 2013.

[CFL06] Yizheng Cai, Nando de Freitas, and James J Little. "Robust visual tracking for multiple targets." In *European conference on computer vision*, pp. 107–118. Springer, 2006.

[CKL17] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. "Visual tracking by reinforced decision making." *arXiv preprint arXiv:1702.06291*, 2017.

[CMP07] Guillaume Charpiat, Pierre Maurel, J-P Pons, Renaud Keriven, and Olivier Faugeras. "Generalized gradients: Priors on minimization flows." *Int. J. Comput. Vis.*, **73**(3):325–344, 2007.

[COL17] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism." In *2017 IEEE International Conference on Computer Vision (ICCV).(Oct 2017)*, pp. 4846–4855, 2017.

[COR16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[CP11] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging." *Journal of Mathematical Imaging and Vision*, 2011.

[CPK17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs." *IEEE transactions on pattern analysis and machine intelligence*, **40**(4):834–848, 2017.

[CZP18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-decoder with atrous separable convolution for semantic image segmentation." In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

[DF18] Emily Denton and Rob Fergus. "Stochastic video generation with a learned prior." *arXiv preprint arXiv:1802.07687*, 2018.

[DGA00] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering." *Statistics and computing*, **10**(3):197–208, 2000.

[DHS15] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. "Learning spatially regularized correlation filters for visual tracking." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4310–4318, 2015.

[Dou98] Arnaud Doucet. "On sequential simulation-based methods for Bayesian filtering." 1998.

[DP91] Trevor Darrell and Alexander Pentland. "Robust estimation of a multi-layered motion representation." In *IEEE Workshop on Visual Motion*, 1991.

[DRK16] Martin Danelljan, Andreas Robinson, Fahad Shahbaz Khan, and Michael Felsberg. "Beyond correlation filters: Learning continuous convolution operators for visual tracking." In *European Conference on Computer Vision*, pp. 472–488. Springer, 2016.

[DWQ17] Dawei Du, Longyin Wen, Honggang Qi, Qingming Huang, Qi Tian, and Siwei Lyu. "Iterative graph seeking for object tracking." *IEEE Transactions on Image Processing*, **27**(4):1809–1821, 2017.

[DZ13] Piotr Dollár and C. Lawrence Zitnick. "Structured Forests for Fast Edge Detection." In *Int. Conf. Comput. Vis.*, 2013.

[Elm90] Jeffrey L Elman. "Finding structure in time." *Cognitive science*, **14**(2):179–211, 1990.

[FPZ17] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Detect to track and track to detect." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3038–3046, 2017.

[GB08] Michael Grant and Stephen Boyd. "Graph implementations for nonsmooth convex programs." In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[GB14] Michael Grant and Stephen Boyd. "CVX: Matlab Software for Disciplined Convex Programming, version 2.1." `http://cvxr.com/cvx`, March 2014.

[GCS12] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. "Video Segmentation with Superpixels." In *ACCV*, 2012.

[GFF18] Daniel Gordon, Ali Farhadi, and Dieter Fox. "Re 3: Re al-time recurrent regression networks for visual tracking of generic objects." *IEEE Robotics and Automation Letters*, **3**(2):788–795, 2018.

[GFZ17] Qing Guo, Wei Feng, Ce Zhou, Rui Huang, Liang Wan, and Song Wang. "Learning dynamic siamese network for visual object tracking." In *The IEEE International Conference on Computer Vision (ICCV).(Oct 2017)*, 2017.

[GKH10a] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. "Efficient Hierarchical Graph Based Video Segmentation." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2010.

[GKH10b] Matthias Grundmann, Vivek Kwatra, Mei Han, and Irfan Essa. "Efficient Hierarchical Graph-based Video Segmentation." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2010.

[GLS13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets Robotics: The KITTI Dataset." *International Journal of Robotics Research (IJRR)*, 2013.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[GNJ13] Fabio Galasso, S. Naveen, Tatiana J. Cardenas, Thomas Brox, and Bernt Schiele. "A Unified Video Segmentation Benchmark: Annotation, Metrics and Analysis." In *Int. Conf. Comput. Vis.*, 2013.

[GPB11] Gottfried Graber, Thomas Pock, and Horst Bischof. "Online 3D reconstruction using convex optimization." In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 708–711. IEEE, 2011.

[Gra12] Alex Graves. "Long short-term memory." *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[GSS93] Neil J Gordon, David J Salmond, and Adrian FM Smith. "Novel approach to nonlinear/non-Gaussian Bayesian state estimation." In *IEE Proceedings F-radar and signal processing*, volume 140, pp. 107–113. IET, 1993.

[GXC19] Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. "Disentangling propagation and generation for video prediction." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9006–9015, 2019.

[HF09] Rob Hess and Alan Fern. "Discriminatively trained particle filters for complex multi-object tracking." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 240–247. IEEE, 2009.

[HGD17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[HGS16] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. "Struck: Structured output tracking with kernels." *IEEE transactions on pattern analysis and machine intelligence*, **38**(10):2096–2109, 2016.

[HHB18] Zekun Hao, Xun Huang, and Serge Belongie. "Controllable video generation with sparse trajectories." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7854–7863, 2018.

[HLH18] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. "Learning to decompose and disentangle representations for video prediction." In *Advances in Neural Information Processing Systems*, pp. 517–526, 2018.

[HLM18] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. "Probabilistic video generation using holistic attribute control." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 452–467, 2018.

[HLR17] Chen Huang, Simon Lucey, and Deva Ramanan. "Learning policies for adaptive tracking with deep feature cascades." In *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 105–114, 2017.

[HLT18] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. "A twofold siamese network for real-time object tracking." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4834–4843, 2018.

[HS81] B.K.P. Horn and B.G. Schunck. "Determining Optical Flow." *Artificial intelligence*, **17**(1-3):185–203, 1981.

[HSA17] Bohyung Han, Jack Sim, and Hartwig Adam. "Branchout: Regularization for online ensemble tracking with convolutional neural networks." CVPR, 2017.

[HTS16] David Held, Sebastian Thrun, and Silvio Savarese. "Learning to track at 100 fps with deep regression networks." In *European Conference on Computer Vision*, pp. 749–765. Springer, 2016.

[HYK15] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. "Online tracking by learning discriminative saliency map with convolutional neural network." In *International Conference on Machine Learning*, pp. 597–606, 2015.

[Iba01] Yukito Iba. "Population monte carlo algorithms." *Transactions of the Japanese Society for Artificial Intelligence*, **16**(2):279–286, 2001.

[JDT16] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. "Dynamic filter networks." In *Advances in Neural Information Processing Systems*, pp. 667–675, 2016.

[JF01] Nebojsa Jojic and Brendan J Frey. "Learning flexible sprites in video layers." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2001.

[JHT20] Beibei Jin, Yu Hu, Qiankun Tang, Jingyu Niu, Zhiping Shi, Yinhe Han, and Xiaowei Li. "Exploring Spatial-Temporal Multi-Frequency Analysis for High-Fidelity and Temporal-Consistency Video Prediction." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4554–4563, 2020.

[JRB18] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. "Differentiable particle filters: End-to-end learning with algorithmic priors." *arXiv preprint arXiv:1805.11122*, 2018.

[JSZ15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks." In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

[JXS17] Xiaojie Jin, Huaxin Xiao, Xiaohui Shen, Jimei Yang, Zhe Lin, Yunpeng Chen, Zequn Jie, Jiashi Feng, and Shuicheng Yan. "Predicting scene parsing and motion dynamics in the future." In *Advances in Neural Information Processing Systems*, pp. 6915–6924, 2017.

[JYS08] J. Jackson, A. J. Yezzi, and S. Soatto. "Dynamic shape and appearance modeling via moving and deforming layers." *Int. J. Comput. Vis.*, 2008.

[Kal60] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems." 1960.

[KBC12] Kalin Kolev, Thomas Brox, and Daniel Cremers. "Fast joint estimation of silhouettes and dense 3D geometry from multiple images." *IEEE Trans. Pattern Anal. Mach. Intell.*, **34**(3):493–505, 2012.

[KBD04] Zia Khan, Tucker Balch, and Frank Dellaert. "An MCMC-based particle filter for tracking multiple interacting targets." In *European Conference on Computer Vision*, pp. 279–290. Springer, 2004.

[KBP17] Adam Kosiorek, Alex Bewley, and Ingmar Posner. "Hierarchical attentive recurrent tracking." In *Advances in Neural Information Processing Systems*, pp. 3053–3061, 2017.

[KLR18] Chanho Kim, Fuxin Li, and James M Rehg. "Multi-object Tracking with Neural Gating Using Bilinear LSTM." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 200–215, 2018.

[KML16] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. "A Novel Performance Evaluation Methodology for Single-Target Trackers." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38**(11):2137–2155, Nov 2016.

[KMM12] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." *IEEE transactions on pattern analysis and machine intelligence*, **34**(7):1409–1422, 2012.

[KMM15] Samira Ebrahimi Kahou, Vincent Michalski, and Roland Memisevic. "RATM: recurrent attentive tracking model." *arXiv preprint arXiv:1510.08660*, 2015.

[Kol84] David A Kolb. "Experience as the source of learning and development." *Upper Sadle River: Prentice Hall*, 1984.

[KSC15] C. Kerl, J. Stueckler, and D. Cremers. "Dense Continuous-Time Tracking and Mapping with Rolling Shutter RGB-D Cameras." In *Int. Conf. Comput. Vis.*, Santiago, Chile, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." *Communications of the ACM*, **60**(6):84–90, 2017.

[KTZ08] M Pawan Kumar, Philip HS Torr, and Andrew Zisserman. "Learning layered motion segmentations of video." *Int. J. Comput. Vis.*, **76**(3), 2008.

[Laf96] Eric Lafortune. "Mathematical models and Monte Carlo algorithms for physically based rendering." *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven*, **20**:74–79, 1996.

[LFY18] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. "Flow-grounded spatial-temporal video prediction from still images." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 600–615, 2018.

[LKC16] William Lotter, Gabriel Kreiman, and David Cox. "Deep predictive coding networks for video prediction and unsupervised learning." *arXiv preprint arXiv:1605.08104*, 2016.

[LKG11] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. "Key-segments for Video Object Segmentation." In *Int. Conf. Comput. Vis.*, 2011.

[LLD17] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. "Dual motion gan for future-flow embedded video prediction." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1744–1752, 2017.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." 2001.

[LNC17] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. "Predicting deeper into the future of semantic segmentation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 648–657, 2017.

[LPH17] Zelun Luo, Boya Peng, De-An Huang, Alexandre Alahi, and Li Fei-Fei. "Unsupervised learning of long-term motion dynamics for videos." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2203–2212, 2017.

[LRS18] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. "Image inpainting for irregular holes using partial convolutions." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 85–100, 2018.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[LYT17] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. "Video frame synthesis using deep voxel flow." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4463–4471, 2017.

[LYW18] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. "High Performance Visual Tracking With Siamese Region Proposal Network." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8971–8980, 2018.

[LZK17] Alan Lukežič, Luka Čehovin Zajc, and Matej Kristan. "Deformable parts correlation filters for robust visual tracking." *IEEE Transactions on Cybernetics*, 2017.

[MHY15] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. "Hierarchical convolutional features for visual tracking." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3074–3082, 2015.

[ML11] Dennis Mitzel and Bastian Leibe. "Real-time multi-person tracking with detector assisted structure propagation." In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 974–981. IEEE, 2011.

[MMB17] Tanya Marwah, Gaurav Mittal, and Vineeth N Balasubramanian. "Attentive semantic video generation using captions." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1426–1434, 2017.

[MRD17] Anton Milan, Seyed Hamid Rezatofighi, Anthony R Dick, Ian D Reid, and Konrad Schindler. "Online Multi-Target Tracking Using Recurrent Neural Networks." In *AAAI*, volume 2, p. 4, 2017.

[MSK04] Yi Ma, Stefano Soatto, Jana Košecká, and Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer, 2004.

[ND10] Richard A Newcombe and Andrew J Davison. "Live dense reconstruction with a single moving camera." In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 1498–1505. IEEE, 2010.

[NH16] Hyeonseob Nam and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2016.

[NRW18] Seyed shahabeddin Nabavi, Mrigank Rochan, and Yang Wang. "Future Semantic Segmentation with Convolutional LSTM." In *BMVC*, p. 137, 2018.

[NZH17] Guanghan Ning, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He. "Spatially supervised recurrent convolutional neural networks for visual object tracking." In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pp. 1–4. IEEE, 2017.

[OMB14] P. Ochs, J. Malik, and T. Brox. "Segmentation of Moving Objects by Long Term Video Analysis." *IEEE Trans. Pattern Anal. Mach. Intell.*, **36**(6), 2014.

[PF13] Anestis Papazoglou and Vittorio Ferrari. "Fast object segmentation in unconstrained video." In *Int. Conf. Comput. Vis.*, 2013.

[PGS14] Silvia L Pintea, Jan C van Gemert, and Arnold WM Smeulders. "Déja vu." In *European Conference on Computer Vision*, pp. 172–187. Springer, 2014.

[PWJ19] Junting Pan, Chengyu Wang, Xu Jia, Jing Shao, Lu Sheng, Junjie Yan, and Xiaogang Wang. "Video Generation From Single Semantic Label Map." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2019.

[Rei79] Donald Reid et al. "An algorithm for tracking multiple targets." *IEEE transactions on Automatic Control*, **24**(6):843–854, 1979.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." *nature*, **323**(6088):533–536, 1986.

[RLS18] Fitsum A Reda, Guilin Liu, Kevin J Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. "Sdc-net: Video prediction using spatially-displaced convolution." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 718–733, 2018.

[RM07] Xiaofeng Ren and Jitendra Malik. "Tracking as Repeated Figure/Ground Segmentation." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2007.

[RSL11] Mikel Rodriguez, Josef Sivic, Ivan Laptev, and Jean-Yves Audibert. "Data-driven crowd analysis in videos." In *ICCV 2011-13th International Conference on Computer Vision*, pp. 1235–1242. IEEE, 2011.

[RWH15] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. "Epicflow: Edge-preserving interpolation of correspondences for optical flow." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1164–1172, 2015.

[SAS17] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. "Tracking the untrackable: Learning to track multiple cues with long-term dependencies." *arXiv preprint arXiv:1701.01909*, **4**(5):6, 2017.

[SBC] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. "Multi-Object Tracking with Quadruplet Convolutional Neural Networks.".

[SBC17] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. "Multi-object tracking with quadruplet convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5620–5629, 2017.

[SC12] Thomas Schoenemann and Daniel Cremers. "A Coding-Cost Framework for Super-Resolution Motion Layer Decomposition." *IEEE Trans. Image Process.*, 2012.

[SCC14] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. "Visual tracking: An experimental survey." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **36**(7):1442–1468, 2014.

[SCW15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." *Advances in neural information processing systems*, **28**, 2015.

[SDC04] P. Smith, Tom Drummond, and R. Cipolla. "Layered Motion Segmentation and Depth Ordering by Tracking Edges." *IEEE Trans. Pattern Anal. Mach. Intell.*, **26**(4), 2004.

[SMG17] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. "Crest: Convolutional residual learning for visual tracking." In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2574–2583. IEEE, 2017.

[SMS17] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. "Temporal generative adversarial nets with singular value clipping." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2830–2839, 2017.

[SR17] James Steven Supancic III and Deva Ramanan. "Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning." In *ICCV*, pp. 322–331, 2017.

[SSB12] Deqing Sun, E.B. Sudderth, and M.J. Black. "Layered segmentation and optical flow estimation over time." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012.

[SSB15] Naveen Shankar Nagaraja, Frank R Schmidt, and Thomas Brox. "Video Segmentation with Just a Few Strokes." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3235–3243, 2015.

[SWS13] Deqing Sun, Jonas Wulff, Erik B Sudderth, Hanspeter Pfister, and Michael J Black. "A fully-connected layered model of foreground and background flow." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2451–2458, 2013.

[SYL18] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8934–8943, 2018.

[TBL19] Adam Terwilliger, Garrick Brazil, and Xiaoming Liu. "Recurrent flow-guided semantic forecasting." In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1703–1712. IEEE, 2019.

[TGS16] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. "Siamese instance search for tracking." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1420–1429, 2016.

[TKS15] Brian Taylor, Vasiliy Karasev, and Stefano Soattoc. "Causal Video Object Segmentation from Persistence of Occlusions." In *IEEE Conf. Comput. Vis. Pattern Recog.* IEEE, 2015.

[TLY18] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. "Mocogan: Decomposing motion and content for video generation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535, 2018.

[VAP10] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. "Multiple Hypothesis Video Segmentation from Superpixel Flows." In *Eur. Conf. Comput. Vis.*, 2010.

[VM17] E Velasco-Salido and JM Martınez. "Scale adaptive point-based kanade lukas tomasi colour-filter tracker." *Under Review*, 2017.

[VMN16] Tomas Vojir, Jiri Matas, and Jana Noskova. "Online adaptive hidden markov model for multi-tracker fusion." *Computer Vision and Image Understanding*, **153**:109–119, 2016.

[VPT16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Generating videos with scene dynamics." In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

[VSR15] Christoph Vogel, Konrad Schindler, and Stefan Roth. "3D Scene Flow Estimation with a Piecewise Rigid Scene Model." *International Journal of Computer Vision*, **115**(1):1–28, 2015.

[VYH17] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. "Decomposing motion and content for natural video sequence prediction." *arXiv preprint arXiv:1706.08033*, 2017.

[VYZ17] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. "Learning to generate long-term future via hierarchical prediction." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3560–3569. JMLR. org, 2017.

[WA94] John Y.A. Wang and Edward H. Adelson. "Representing Moving Images with Layers." *IEEE Trans. Image Process.*, 1994.

[WBS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. "Image quality assessment: from error visibility to structural similarity." *IEEE transactions on image processing*, **13**(4):600–612, 2004.

[WGP20] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. "Future video synthesis with object motion prediction." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5539–5548, 2020.

[WLZ18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "Video-to-video synthesis." *arXiv preprint arXiv:1808.06601*, 2018.

[WMG17] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. "The pose knows: Video forecasting by generating pose futures." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3332–3341, 2017.

[WOW15] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. "Visual tracking with fully convolutional networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3119–3127, 2015.

[WP90] Ronald J Williams and Jing Peng. "An efficient gradient-based algorithm for on-line training of recurrent network trajectories." *Neural computation*, **2**(4):490–501, 1990.

[WSB03] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. "Multiscale structural similarity for image quality assessment." In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pp. 1398–1402. Ieee, 2003.

[WSY17] Wenguan Wang, Jianbing Shen, Ruigang Yang, and Fatih Porikli. "Saliency-aware video object segmentation." *IEEE transactions on pattern analysis and machine intelligence*, **40**(1):20–33, 2017.

[WTX18] Qiang Wang, Zhu Teng, Junliang Xing, Jin Gao, Weiming Hu, and Stephen Maybank. "Learning attentions: residual attentional Siamese Network for high performance online visual tracking." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4854–4863, 2018.

[WVE18] Nevan Wichers, Ruben Villegas, Dumitru Erhan, and Honglak Lee. "Hierarchical long-term video prediction without supervision." *arXiv preprint arXiv:1806.04768*, 2018.

[WY13] Naiyan Wang and Dit-Yan Yeung. "Learning a deep compact image representation for visual tracking." In *Advances in neural information processing systems*, pp. 809–817, 2013.

[XCW15] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." In *Advances in neural information processing systems*, pp. 802–810, 2015.

[XGY17] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. "End-to-end learning of driving models from large-scale video datasets." *arXiv preprint*, 2017.

[XNL18] Jingwei Xu, Bingbing Ni, Zefan Li, Shuo Cheng, and Xiaokang Yang. "Structure preserving video prediction." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1460–1469, 2018.

[XWB16] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks." In *Advances in neural information processing systems*, pp. 91–99, 2016.

[XXC12] Chenliang Xu, Caiming Xiong, and Jason J. Corso. "Streaming Hierarchical Video Segmentation." In *Eur. Conf. Comput. Vis.*, 2012.

[XYW11] Jiang Xu, Junsong Yuan, and Ying Wu. "Learning spatio-temporal dependency of local patches for complex motion segmentation." *Computer Vision and Image Understanding*, **115**(3):334–351, 2011.

[YC17] Tianyu Yang and Antoni B Chan. "Recurrent filter learning for visual tracking." *arXiv preprint arXiv:1708.03874*, 2017.

[YCS20] Yanchao Yang, Yutong Chen, and Stefano Soatto. "Learning to manipulate individual objects in an image." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567, 2020.

[YDD05] Changjiang Yang, Ramani Duraiswami, and Larry Davis. "Fast multiple object tracking via a hierarchical particle filter." In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pp. 212–219. IEEE, 2005.

[YLS19] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. "Unsupervised moving object detection via contextual information separation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 879–888, 2019.

[YS13] Yanchao Yang and Ganesh Sundaramoorthi. "Modeling Self-Occlusions in Dynamic Shape and Appearance Tracking." In *Int. Conf. Comput. Vis.*, 2013.

[YSS15] Yanchao Yang, Ganesh Sundaramoorthi, and Stefano Soatto. "Self-Occlusions and Disocclusion in Causal Video Object Segmentation." In *Int. Conf. Comput. Vis.*, 2015.

[YWZ18] Ceyuan Yang, Zhe Wang, Xinge Zhu, Chen Huang, Jianping Shi, and Dahua Lin. "Pose guided human video generation." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 201–216, 2018.

[ZGH16] Shun Zhang, Yihong Gong, Jia-Bin Huang, Jongwoo Lim, Jinjun Wang, Narendra Ahuja, and Ming-Hsuan Yang. "Tracking persons-of-interest via adaptive discriminative features." In *European Conference on Computer Vision*, pp. 415–433. Springer, 2016.

[ZIE18] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The unreasonable effectiveness of deep features as a perceptual metric." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

[ZJS13] Dong Zhang, Omar Javed, and Mubarak Shah. "Video Object Segmentation through Spatially Accurate and Temporally Dense Extraction of Primary Object Regions." In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2013.

[ZMW17] Da Zhang, Hamid Maei, Xin Wang, and Yuan-Fang Wang. "Deep Reinforcement Learning for Visual Object Tracking in Videos." *arXiv preprint arXiv:1701.08936*, 2017.

[ZPB07] C. Zach, T. Pock, and H. Bischof. "A duality based approach for realtime TV-L 1 optical flow." *Pattern Recognition*, pp. 214–223, 2007.

[ZPI17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks." In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[ZPT18] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris Metaxas. "Learning to forecast and refine residual motion for image-to-video generation." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 387–403, 2018.