Learning From Multi-Frame Data

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Patrick Wieschollek

aus Erfurt

Tübingen
2019

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	29.07.2020
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Hendrik P. A. Lensch
2. Berichterstatter:	Prof. Dr. Andreas Geiger

Making predictions is very difficult — especially without data.

Abstract

Multi-frame data-driven methods bear the promise that aggregating multiple observations leads to better estimates of target quantities than a single (still) observation. This thesis examines how data-driven approaches such as deep neural networks should be constructed to improve over single-frame-based counterparts. Besides algorithmic changes, as for example in the design of artificial neural network architectures or the algorithm itself, such an examination is inextricably linked with the consideration of the synthesis of synthetic training data in meaningful size (even if no annotations are available) and quality (if real ground-truth acquisition is not possible), which capture all temporal effects with high fidelity.

We start with the introduction of a new algorithm to accelerate a nonparametric learning algorithm by using a GPU adapted implementation to search for the nearest neighbor. While the approaches known so far are clearly surpassed, this empirically reveals that the data generated can be managed within a reasonable time and that several inputs can be processed in parallel even under hardware restrictions. Based on a learning-based solution, we introduce a novel training protocol to bridge the need for carefully curated training data and demonstrate better performance and robustness than a non-parametric search for the nearest neighbor via temporal video alignments. Effective learning in the absence of labels is required when dealing with larger amounts of data that are easy to capture but not feasible or at least costly to label.

In addition, we show new ways to generate plausible and realistic synthesized data and their inevitability when it comes to closing the gap to expensive and almost infeasible real-world acquisition. These eventually achieve state-of-the-art results in classical image processing tasks such as reflection removal and video deblurring.

Kurzfassung

Datengesteuerte Verfahren, welche auf Multi-Bild-Eingaben basieren können durch die Aggregation mehrerer Beobachtungen im Vergleich zu ihren Einzelbild Varianten zu einer besseren Schätzung der Zielgröße führen. Diese Arbeit untersucht, wie tiefe künstliche neuronale Netze als datengesteuerte Ansätze konstruiert werden sollten, um sich gegenüber den Einzelbild-basierten Pendants zu verbessern. Neben den algorithmischen Veränderungen in dem Design von künstlichen neuronalen Netzwerkstrukturen oder dem Trainingsalgorithmus selbst ist eine solche Untersuchung untrennbar von der Betrachtung der Datengenerierung von künstlichen Trainingsdaten. Diese Betrachtung muss sowohl einen sinnvollem Umfang der Trainingsdaten als auch eine aussagekräftiger Qualität (z.B. wenn Ground Truth Aufnahmen unmöglich sind) umfassen. Die Generierung von Trainingsdaten für das Lernen von Multi-Bild-Eingaben bringt zusätzliche Herausforderungen, wie temporale Effekte durch dynamische Szenen, falsch ausgerichtete Aufnahmen und dem nicht zu vernachlässigen Speicherbedarf.

Diese Arbeit führt einen neuen Algorithmus zur Beschleunigung eines nichtparametrischen Lernalgorithmus ein. Dieser basiert auf der Verwendung einer GPU-angepassten Implementierung zur Suche nach dem nächsten Nachbarn in großen Datenmengen. Es zeigt sich, dass unter vergleichbarer Qualität zu bisherigen Ansätzen, diese in Ausführungsgeschwindigkeit deutlich übertroffen werden. Damit können nun auch große Datensätzen in plausibler Zeit verarbeitet werden – auch wenn die Hardware bestimmte Einschränkungen bereithält. Da das Annotieren bei Trainingsdaten meist zeitaufwendig, nicht praktikabel oder gar unmöglich ist, stellt diese Arbeit ein neuartiges Trainingsprotokoll vor, um den Bedarf an sorgfältig annotierten Trainingsdaten zu umgehen. Der Ansatz zeigt, dass ein effektives Lernen auf neuronalen Netzen ohne Annotationen möglich ist und auf größere Datenmengen skaliert, wie dies später am Beispiel der Videosynchronisierung gezeigt wird.

Darüber hinaus zeigt diese Arbeit neue Wege auf, um plausible und realistische Trainingsdaten für klassische Computer-Vision-Probleme synthetisch zu erzeugen. Es ist unvermeidlich einen Anspruch auf solche Genauigkeit zu stellen, um die Lücke zwischen synthetisch simulierter und real aufgenommener Daten zu schließen und damit die Generalisierung der tiefen neuronale Netzen zur Inferenz sicherzustellen. Dies wird an Beispielen, wie Entfernung von Reflexionen in Bildern und Unschärfe in Videos, vorgestellt. Dabei zeigt sich, dass die Verbindung zwischen synthetischen Trainingsdaten, mit hoher Detailtreue zu realistischen Daten, und angepasste datengetriebenen Algorithmen zu deutlich besseren Ergebnissen führt.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Hendrik P.A. Lensch, for his outstanding guidance. I would like to particularly thank him for his technical help, his original ideas, his precious advice on scientific working, and insightful discussions. I appreciate the opportunity he and Bernhard Schölkopf gave me to pursue my research. I would like to thank Andreas Geiger for agreeing to serve as an reviewer for this thesis at short notice.

I also have a thought for Michael Hirsch and my colleagues at the graphics group at the University of Tübingen and Max Planck Institute, without whom these last years would not have been the same. Thank you for creating an intellectually stimulating environment, thank you for your friendship, and thank you for the intensive discussions. Thanks to Fabian Groh, Katharina Schwarz, Benjamin Resch, Matthias Bauer, Mehdi S. M. Sajjadi, Behzad Tabibian, Raphael Braun, Mark Boss, Dennis Bukenberger, Sebastian Herholz, Jieen Chen, and Arijit Mallick. On a more personal level, I would also like to thank them for their human qualities. The process of research inevitably entails moments of doubt and, sometimes, seemingly insurmountable obstacles. And I was fortunate being able to could count on their support during these moments – even when deadlines were late into the night of some submissions.

I owe a special thanks to Orazio Gallo and Jinwei Gu for being such great mentors during my internship at NVIDIA. It was a great experience working with them.

Thanks to all my collaborators for having provided an important contribution to the development of several projects.

Lastly, these acknowledgments would not be complete without a word of thanks for my family, who gave me support over these years. They offered a sympathetic ear when I needed, and I am convinced that I could not have come this far without their continuous support throughout my life.

Papers Included In This Thesis

Papers, which are included in this thesis:

[Wie+16a] Patrick Wieschollek, Oliver Wang, Alexander Sorkine-Hornung, and Hendrik P.A. Lensch. „Efficient Large-scale Approximate Nearest Neighbor Search on the GPU“. in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. DOI: 10.1109/CVPR.2016.223

[WFL17] Patrick Wieschollek, Ido Freeman, and Hendrik P. A. Lensch. „Learning Robust Video Synchronization without Annotations“. In: *IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2017. DOI: 10.1109/ICMLA.2017.0-173

[Wie+16b] Patrick Wieschollek, Bernhard Schölkopf, Hendrik P. A. Lensch, and Michael Hirsch. „End-to-End Learning for Image Burst Deblurring“. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2016. DOI: 10.1007/978-3-319-54190-7_3

[Wie+17] Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, and Hendrik P. A. Lensch. „Learning Blind Motion Deblurring“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017. DOI: 10.1109/ICCV.2017.34

[Wie+18] Patrick Wieschollek, Orazio Gallo, Jinwei Gu, and Jan Kautz. „Separating Reflection and Transmission Images in the Wild“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018

Papers, which are related to, but not included in this thesis:

[WGL17] Patrick Wieschollek, Fabian Groh, and Hendrik P. A. Lensch. „Backpropagation Training for Fisher Vectors within Neural Networks“. In: *CoRR* (2017)

[GWL18] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. „Flex-Convolution (Deep Learning Beyond Grid-Worlds)“. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2018

I would like to thank my co-authors for their permission to base chapters of my thesis on joint publications.

Contents

Rule #4:
A deep neural network will cheat whenever possible!

Chapter 1

Introduction

The collection of an enormous amount of data has become a lot easier in recent years. By developing new hardware for more cost-effective storage and faster computation, data-driven approaches - and in particular deep artificial neural networks - can uncover concealed patterns and structures within the collected and merely unstructured data and gain additional knowledge and insights. This enables remarkable technologies such as end-to-end autonomous driving [Boj+16], machine translation [Vas+17], text-to-speech [Hsu+14] and image classification [He+16a] or generation [AB19] to name just a few we were allowed to witness in the last few years. Nevertheless, there are fundamental limits to the success of these techniques. These limitations range from more technical limitations such as computational demands and memory consumption, *e.g.* when analyzing minute-long video streams, to theoretical challenges such as generalizing over longer sequences that have not been trained due to the hardware limitations described above.

The majority of tasks where the latest machine learning methods have reached the state-of-the-art shifted the focus from traditional feature engineering to data engineering. Undoubtedly, one of the primary ingredients for the early success of accurate object classification, *e.g.* ImageNet-Challenge [Den+09], is the availability of a carefully curated set of human annotations in reasonable quantities. However, the development of data-driven methods capable of *tabula rasa* learning without any human annotation is necessary if these methods are to be applied to problems where annotation of data is time-consuming, costly, or even impracticable. Consider online platforms that allow users to share hours of video content. While these sources provide comprehensive data on our environment and everyday life, they are rarely labeled and can therefore hardly be used as a direct learning signal in the application of data-driven approaches. Transfer-learning in combination with fine-tuning may not be the solution for a variety of tasks, even if only a small part of the available labels is missing. This leads to requirements for surrogate models and novel iterative training protocols, which collect more and more useful training data over time fully automatically.

The development of such methods, which can operate without the need for anno-

tations, appears to be problematic from a theoretical point of view concerning classical back-propagation algorithms and commonly used loss functions. In recent years, the trend of AlexNet [KSH12], VGG [SZ14], Inception [Sze+15], ResNet [He+16b] and DenseNet [Hua+17] had only the direction of deeper models with more layers and improvements in the choice of architectural design or creating larger non-public datasets [HVD15]. The training protocol, however, remained unaltered. Further, the use of multiple data sources – possibly unlabeled – has not yet become standard practice. This is inexplicable, as annotated data is rarely or at best costly to obtain. Moreover, it seems to be a rather unnatural way of learning. Even though we were never given an exact definition (label) of the term "clockwise", we still understand it as a rotation of an object in the well-defined direction given by the clock hands. Understanding this kind of abstraction by leveraging our experience (how the clock hands turn over time) and combining it with certain linguistic concepts (compound words) helps us to navigate successfully through our environment and make sense out of multiple sensorial impressions. Multi-frame information as several successive sensory inputs underlying the constraint of temporal coherence (arrow of time) can thus convey a strong learning signal — even in the absence of annotations.

There is another motivation considering multi-frame methods. The landscape in how we interact with photography has changed dramatically in recent years, be it as a casual photographer or in a profession. However, the development of the photography from the "Camera Obscura" to the latest built-in sensors in smartphones can not only be understood from the perspective of the hardware development over 35mm film or single-lens reflex cameras (SLRs). Since the arrival of digital capturing devices like MegaVision Tessera[1] in 1987, improving the capturing process is not anymore purely based on better optics hardware. Instead, the digital data format allowed complex post-processing pipelines, starting with raw denoising, automatic white balancing, photography of high dynamic range (HDR) and is far from over with synthesizing depth of field effects [Wad+18] in an "app". Eventually, a single deep neural network DeepISP [SGB19] handles all the individual steps involved in traditional image-signal-processing (ISP) pipelines.

Further circumstances, like the spherical aberration on the Hubble Space Telescope caused by the mounted mirror of the device [Whi92] led to several algorithm [Lin17; Las90] to restore sharp observations before corrective optics has been installed on site[2]. Popular attempts of today's computer vision target ill-posed problems such as the removal of obstructions by in-painting these regions from hallucinated plausible content [Liu+18] or the increase of the spatial resolution in photos [Lai+17], removal of unwanted reflections [LB13] or handling blur due to camera shake [Cha16] using deep neural networks. In the past, whenever a tripod was necessary in order to obtain a relatively sharp photograph, these aids

[1] `http://www.mega-vision.com/why_Measured_Photography.html`
[2] `https://www.nasa.gov/content/hubbles-mirror-flaw`

have become increasingly superfluous and can almost be replaced by calculations. Recently it has been shown that neural networks are capable of hallucinating entire high-resolution image contents from noise and massive data [AB19; Kar+18] using generative adversarial networks (GANs). Therefore, it is questionale to use methods like GANs to recover galaxy features from mediocre observations [Sch+17] whenever accurate observations are necessary but training data is rare.

Now, as the development of hardware has not stopped, sensors are capable of recording *multiple* images at high frame-rates. Even in smartphones, multiple frames are taken to deal with noise in dimly lit scenes as quite recently done in Google's "Night Sigth"[3]. Most intriguing, our own perception – based on astonishing hardware (the human eye) – is delivering 10Mbps continuous data transmission [Koc+06] on the way to our brain. This is the throughput a common Ethernet connection is able to forward. When millions of years in evolutionary opportunity to come up with alternative perception systems still converges to such a system, it is evidently prodigal to drop almost all sensory impressions and temporal context by just using a single observation in an artificial system, which aims at reproducing human intelligence capabilities. Instead, it is standing to reason to follow the same practices evolution came up with and feed respectively handle a continuous stream of observations in data-driven approaches — without any explicit learning signal from an external teacher.

Although multiple observations *might* promise better estimates if they are independent, they also pose additional challenges such as the temporal and spatial (mis-)alignment of different observations. And they seize greedily computational resources as the input is given an additional dimension: "Time". As the effects in real-world practice become more and more difficult to grasp, a pipeline for synthetic data generation also becomes more complicated compared to single frame based counterparts. To pre-empt a later chapter of this work: The synthesis of blur in images caused by camera shake is well understood in literature and mold in a relatively simple mathematical model. However, the temporal multi-frame counterpart, which deals with dynamic content, requires a novel and far more sophisticated approach to generate plausible synthetic training data at scale. Finally, this shapes two forms of inputs for multi-frame methods which we are going to investigate in this thesis: those that are observed under different circumstances (*e.g.* different polarization of light) without a particular order and methods that stream data directly into a neural network with a canonical order — both are potentially spatially misaligned.

[3] `https://ai.googleblog.com/2018/11/night-sight-seeing-in-dark-on-pixel.html`

1.1 Main Contributions and Research Questions

The primary objective of this thesis is to design deep learning based models to handle multi-frame data more naturally. This presents two challenges: the generation of data of appropriate size and quality and the handling of data in learning-based algorithms.

As a recurring theme, all the deep-learning based approaches presented in this thesis share the ability to generate almost infinite amounts of training data with various distortions *automatically* to optimize capable models that eliminate these unwanted effects — even if there is no ground-truth available. Fortunately, for most problems (*e.g.* deblurring, removing reflections) the inverse problem (*e.g.* blurring a photo, adding reflections) once defined is somewhat easier to accomplish and its effect easier to synthesize. A second recurring pattern is the faithful replication of real-world data in a synthesized training set in which all effects such as blur and reflections are captured with high fidelity. In all chapters, we further show how more plausible and realistic synthesized multi-frame data can close the gap to an expensive and almost unfeasible real-world acquisition, leading to better models and estimates that ultimately come closer to solving these problems.

Still, handling entire data streams requires algorithms which work at acceptable costs in terms of time and memory. This is likewise true for rather classical approaches such as nearest neighbor search, and it becomes the main focus later in this thesis when assembling approaches based on neural networks.

In detail, we formulate the research questions and main contributions as:
Research Question 1: *Can we utilized modern hardware like the Graphics Processing Unit (GPU) to gain a speed-up over traditional CPU based nearest neighbor approaches without sacrificing recall-performance?*

The nearest neighbor problem as a non-parametric data-driven method is a central part of traditional computer vision pipelines and typically represents a massive bottleneck. GPU hardware fueled latest developments in deep learning. Therefore, it is important to identify other classical problems that may profit from recent breakthroughs in hardware development. We introduce the first GPU approach obtaining a significant speed-up over all previous methods [Wie+16a] and at the same time improving the accuracy of previous attempts.

Research Question 2: *Can we train a neural network to understand video content from a continuous data stream without any external training signal?*

We introduce a novel training protocol [WFL17] for *tabula rasa learning* for robust video synchronization beyond the field of reinforcement algorithms for deep-learning. Hereby, our method iteratively increases the complexity of the training data autonomously. This is done by a iterative process which based on exploiting temporal coherence constraints in videos to train a deep neural network over several iterations. The current iteration of a trained deep neural network is used

to gather training examples via predictions which might be missed by a previous (less-trained) version the neural network. Hereby, false positives and false negatives are rejected based on a novel heuristic to avoid contamination of the next generation of the gathered training data. The newly assembled training data is used to refine the deep neural network. The final trained model can synchronize videos, which are even months apart and their appearances almost do not resemble each other [WFL17].

Research Question 3: *Given there is no practical way to reliable gather ground-truth data, can we still use a data-driven learning method to, for example, remove reflection from images?*

The removal of reflections is vital when it comes to estimating correspondences in multi-frame computer vision methods. Ambiguities caused by reflection are typically instances where such an algorithm fails. We have developed a novel image synthesis pipeline [Wie+18] that faithfully reproduces reflections and the polarization effects from real examples with an imaging-based rendering. This resulted in state-of-the-art results on multi-frame reflection removal in-the-wild, even when based on a rather small neural network architecture to ensure this model fits hardware constraints on todays smartphones.

Research Question 4: *How can we merge classical multi-frame approaches and deep neural networks to further improve the neural network performance?*

While most learning-based methods work entirely independently to classical approaches, we have modified a classical lucky-imaging method to integrate it as a novel layer into a deep neural network. We further demonstrate the combination of both can be trained and a joint-training leads to better results [Wie+16b] than a combination of several approaches.

Research Question 5: *Given a stream of continuous video data, how can a deep-learning approach handle arbitrary long sequences while having low memory and computational footprint?*

Handling the additional time dimension poses some challenges. While typical 2D convolutional layers are translation invariant, existing methods with image inputs based on recurrent cells [PHC16] are difficult to train or not flexible enough to handle different sequence lengths [Su+17]. We propose a novel way of recurring network blocks to deal with a stream of data in a neural network [Wie+17]. This obtains state-of-the-results in video deblurring and generalizes well to sequences of lengths which are never observed during training. Further, the specific network layout design is adjusted such that it can process full HD-frames even on modest GPU-hardware.

1.2 Outline

The outline of this thesis is as follows: Chapter 2 introduces a practical and efficient approach to solve the Approximate Nearest Neighbor (ANN) problem in a large-scale setting. Efficiently finding related points in a database for a given query point using ANN states an integral part in many relevant computer vision approaches. While ANN algorithms previously remained as a critical bottleneck in these approaches, we introduce the first method utilizing the parallelism of GPU devices to provide superior performance compared to previous attempts.

Such an ANN approach can be used to align entire video sequences temporally or for loop-closing in Simultaneous Localization and Mapping (SLAM) approaches. However, this introduces two challenges: While matching multiple local descriptors, *e.g.* Scale-invariant Feature Transform (SIFT) vectors, is possible (similar to Chapter 2), learning a single scene-descriptor would alleviate the impact of the nearest neighbor lookup speed by employing a single global descriptor. Further two videos showing the same content might feature drastic appearance differences, *e.g.* consider out-door videos captured during summer and winter. A global descriptor might be capable of encoding only relevant information, but inherits all requirements from data-driven approaches — mainly the demand for accurately annotated data. To circumvent these issues, we present a *tabula rasa* learning approach based on Convolutional Neural Networks (CNNs) in Chapter 4 to automatically analyze and temporally align videos which are eventually recorded months apart sharing the same scene under different illumination, seasonal effects and motion blur created by the recording equipment. Importantly, the presented method autonomously manages the underlying training data for learning a meaningful representation in an iterative procedure collecting training examples with increased complexity from a large corpus of un-annotated videos. Consequently, it removes the need of human intervention in the training loop at all.

In contrast to a single global 2D scene descriptor (Chapter 4), common 3D reconstructions methods seek for a per-pixel depth annotation. However, when working with images or videos acquired in the wild one has to deal with different effects reducing the image quality or even violating commonly made assumptions. The remaining two Chapters 5, 6 address the problem of restoring the unobserved ground-truth image from observations deteriorated by different causes.

Chapter 5 deals with the problem of separating reflection and transition images. It is virtually impossible to avoid semi-reflectors in human-made environments. Hence, as they can severely impact the performance of computer vision algorithms, *e.g.* during computing correspondences matching (required in dense 3D reconstructions) it is required to disambiguate such super-imposed information. Instead of using temporal information (Chapter 4), we describe a multi-frame approach which exploits physical properties of natural light in the presence of different polarization states. Modeling the physics of light (polarization on thin glass plates), imperfect-

ness of semi-reflectors (geometry of scene) and scene dynamics (object motion) in an image-based training data generation allows for separating reflections and transmissions in images later captured in common places using a convolutional neural network.

Finally, while approaches relying on a temporal sequence like videos provide additional insights, *e.g.* consider the ambiguity between "sit down" and "stand up" from a seat in a single shot. However, these multi-frame inputs capturing different appearances (*e.g.* different polarization states, Chapter 5) are confronted with additional challenges like spatial alignment or blur. Chapter 6 therefore addresses the problem of propagating temporal information across multiple frames to reduce blur in the recorded sequences. While blur caused by camera shake (ego-motion) can be modeled and synthesized in a relatively simple mathematical model, simulating respectively removing motion blur (object-motion) from a dynamic scene challenges any method. It requires to broadcast extracted information from one time step to consecutive time steps — possibly over completely different spatial locations to successfully restore the sharp ground-truth frame.

Rule #3:
More data has never been harmful.

Chapter 2

Nearest Neighbor Search

Given a large collection of data points in a potentially high-dimensional space, finding the nearest neighbor for a particular query is a serious problem in image processing. While previous attempts either accept loss in speed due to a comprehensive search on the GPU or perform this search on the CPU due to sequential algorithm components, we demonstrate the first method that is specifically tailored to the GPU architecture and offers superior performance.

The material of this chapter is based on the following publication:

[Wie+16a] Patrick Wieschollek, Oliver Wang, Alexander Sorkine-Hornung, and Hendrik P.A. Lensch. „Efficient Large-scale Approximate Nearest Neighbor Search on the GPU“. in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. DOI: 10.1109/CVPR.2016.223

Consider a set $\mathcal{X}$ of vectors $x \in \mathbb{R}^d$ in Euclidean space with metric $\delta(a, b) = \|a - b\|_2$ representing collected data. Hereby, commonly used dimensions are $d = 128$ for SIFT vectors or $d \geq 1024$ for extracted features from a deep convolutional neural network such as ResNet [He+16a]. Finding the nearest neighbor $N(q) \in \mathcal{X}$ of a query vector $q \in \mathbb{R}^d$ in such high-dimensional space is a fundamental task in computer vision that can be formulated as

$$N(q) := \arg\min_{x \in \mathcal{X}} \delta(q, x). \tag{2.1}$$

However, extracting and collecting SIFT features from common scenes such as shown in Figure 2.1 typically results in 20k local features for each single frame. Hence, building such a dataset for a 30-minute short video clip typically produces $n > 10^9$ data points with the ensuing need of pairwise matching. In practice d as well as n are often large, leading to nearest neighbor searches being a significant computational bottleneck in many applications when relying on a dense correspondence matching. This is due to the necessity of performing exact distance

Figure 2.1: Extracting local features such as SIFT from a fairly simple scene (left) frequently results in 20k features per frame (right). Any dense correspondence matching for multiple frames algorithm would be hard stressed without a reasonable acceleration structure for the nearest neighbor search component.

computations in a high-dimensional space between many pairs of vectors, a problem exacerbated by the phenomenon known as the *curse of dimensionality*.

This phenomenon can be easily illustrated by scaling $\mathcal{X}$ to fit into an enclosing d-dimensional hyper-unit cube. Exploring a v fraction of the volume to identify potential nearest neighbor candidates requires to visit at least a $v^{1/d}$ percent of *each* hyper-cube edge. Hence, to explore 10% of a SIFT vectors set ($d = 128$) in a hyper unit-cube, one has to potentially search an interval covering $\approx 98\%$ of the possible values *per* coordinate. Also, the use of the advantages of parallel computations on GPU devices by processing batches of multiple queries simultaneously does not resolve this dilemma.

While leveraging GPU parallelism seems obvious, in practice accelerating nearest neighbor search techniques using GPU parallelism is notoriously tricky, mainly due to hardware constraints, *e.g.* memory restrictions of GPUs compared to the amount of RAM available for CPU-based methods, and algorithmically constraints, *e.g.* sequential nature of the local search. For a concrete example, consider an NVIDIA Titan X GPU, which provides 12GB addressable global memory compared to workstations providing 512GB of RAM. As a result, previously existing GPU-based methods often either exploit the parallelism by implementing exhaustive search approaches, which are limited to small datasets of up to 225 candidate neighbors [Tsa+14] to fit data into memory or handle only 3-dimensional vectors [ML14], making these approaches unsuited for many vision problems. A highly optimized exhaustive search on 1 Million SIFT vectors performed on the NVIDIA Titan X GPU – providing perfect accuracy – takes 23ms per query compared to 5.32ms from a CPU-based approximate nearest neighbor approach like FLANN [ML14].

Hence, accepting a minimal loss in accuracy enables the usage of vector compression methods to provide a significant speed-up. Relaxing the computational

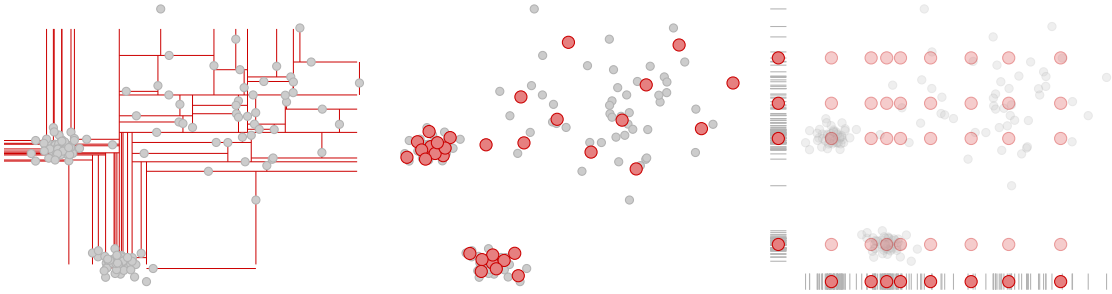
complexity is desired, as for most applications, an *approximate* nearest neighbor search is sufficient, which seek to retrieve the nearest neighbor for a given query "only" with a high probability. Designing such a compression technique in combination with a GPU-tailored index structure is advantageous for such rather memory-limited devices, *e.g.* storing a plain index structure itself already occupies $\approx$ 8GB of memory — assuming each such datapoint ($N = 10^9$) representation is limited to be represented by at most 8 bytes.

2.1 Related Work

There exist many sequential CPU-approaches for computing ANN in the literature, the most common of which are *KD-trees* [FBF77], which hierarchically subdivide the vector space iteratively in each axis. While these methods are widely used in graphics and vision, it has been shown that KD-trees are no more efficient than brute force searches when d is large [JDS11]. The FLANN software package [ML14] proposes randomized KD-forests and k-Means trees, which prune the overall search space by identifying small regions around the query vectors, yielding better performance with higher dimensional vectors.

Another family of approaches is based on the idea of *Locality Sensitive Hashing* (LSH) [Dat+04]. These methods hash database vectors with a number of random projections and perform nearest distance checks only on vectors that are hashed to the same bin. The speed and accuracy of such methods depend on the hashing function used. Andoni and Indyk [AI08] describe a family of hashing functions which are near-optimal. These ideas have since been used in the computer vision community by extending them into the image domain for patch-based nearest neighbor computation [KA11]. While these methods work well, they have not yet achieved the same performance as space partitioning methods [ML14].

One commonly used concept for nearest neighbor search [JDS11; KA14; ML14; Ge+13; BL12] is using two different phases: The *offline phase* is used to build an index structure for accelerating a query lookup relying only on available information from the data set representing the search space. Such a structure is optimized to represent global as well as local information such that, a query (during the *online phase*) can pin down the search space and enumerate datapoints in the vicinity. The *online phase* can exploit query-dependent information in combination with the prebuilt index structure. A common strategy in the *offline phase* is to balance the trade-off between reducing the search space by grouping multiple dataset entries into bins and the overhead of maintaining these bins during the *online phase* when only addressing a few candidate bins during a query. A recurrent pattern is the maintenance of priority queues guiding the traversal of bins, which contain multiple candidate vectors. Unfortunately, updating or synchronizing a priority queue per CUDA kernel is notoriously challenging due to its sequential nature and indepen-



a) KD-tree b) Vector-Quantization (VQ) c) Product-Quantization (PQ)

Figure 2.2: Illustration of different quantization methods for accelerating a nearest neighbor query on a toy dataset. The KD-tree illustration results from a few iterations for clarity. In PQ, the clustering is done in sub-spaces (here each axis) and then re-projected.

dence of parallel launched CUDA blocks when not using atomic operations or a device-wide synchronization. Hence, a GPU-based method — like ours — needs to relax such constraints. A successor of our work, FAISS [JDJ17], even further decreases the required query-time by optimizing the GPU usage mainly by relying on in-register memory and kernel fusing. Additionally, they showed increased performance when employing multiple GPU devices. Apparently, such a scaling gives superior throughput on the GPU at the cost of the query latency. Quite recently, there has been some interest in exploring alternative hardware like the Intel HARPv2 FPGA platform [ZKL18].

Vector Quantization (VQ) [LBG80] is a simple compression method that *clusters* the search space into some bins based on the distance to the cluster centroid. If a query vector is quantized to a bin, all other vectors in that bin are likely to be good candidates for being the nearest neighbor. Unfortunately, if a query lies at the edge of a bin, one has to consider all neighboring bins as well, and the number of neighbors to each Voronoi cell increases *exponentially* w.r.t to the dimension D of the space.

The concept of Product Quantization (PQ) was introduced by Sik [KS99] and made popular in the computer vision community by Jégou et al. [JDS11]. Several state-of-the-art ANN approaches extend these ideas, such as locally optimized product quantization[KA14] and the inverted multi-index [BL12]. These methods currently provide the most efficient techniques for ANN search for high-dimensional data, in terms of speed, accuracy, and memory requirements. As our introduced Product Quantization Tree (PQT) represent an extension to the family of PQ methods, we will describe PQ in more detail in the next section.

2.1.1 Vector and Product Quantization

Our approach builds on the idea of Vector Quantization VQ and Product Quantization (PQ) [JDS11], which we describe in the following section.

Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite set of database vectors $x_i \in \mathbb{R}^d$. Without loss of generality we consider the Euclidean space (X, δ) with $\delta(a, b) = \|a - b\|_2$. However, these approaches can be used with any arbitrary metric δ.

In *Vector Quantization* (VQ), each vector $x \in \mathbb{R}^d$ is encoded by a codebook $\mathcal{C} = \{c_1, \ldots, c_k\}$ of k centroids using the *discretization*-mapping:

$$c \colon \mathcal{X} \to \mathcal{C}, \quad x \mapsto c(x) := \arg\min_{c \in \mathcal{C}} \|x - c\|_2 . \tag{2.2}$$

In other words, each vector x is represented by its closest centroid c_j in the codebook. The set $\mathcal{C}_j = \{x \in \mathbb{R}^d \mid c(x) = c_j\}$ containing all vectors x which are mapped to c_j is called the *cluster* or *bin* for centroid c_j. This quantization of vectors introduces an approximation error $\|x - c(x)\|_2$, but allows for quick retrieval of a set $\mathcal{C}_j$ of similar vectors, i.e., all those vectors from $\mathcal{X}$ that are quantized to the same bin like the query. Classical Lloyd iterations [Llo06] can be used on a subset of the original data to approximate a good codebook $\mathcal{C}$ efficiently. For each query q finding a bin from a codebook of k bins requires k full vector comparisons having complexity $\mathcal{O}(d \cdot k)$.

In *Product Quantization* (PQ), the high-dimensional vector space $\mathbb{R}^d$ is decomposed as a Cartesian product

$$\mathbb{R}^d = \underbrace{\mathbb{R}^m \times \mathbb{R}^m \times \ldots \times \mathbb{R}^m}_{p \text{ times}} \tag{2.3}$$

of p parts, when $d = p \cdot m$.

Likewise, each vector $x \in \mathbb{R}^d$ can be written as a concatenation of p vector-parts $x = ([x]_1, [x]_2, \ldots, [x]_p)^\top$ with $[x]_j \in \mathbb{R}^m$. In Product Quantization, each subspace $\mathbb{R}^m$ is then quantized independently using VQ. This allows for exponentially large codebook of k^p bins by encoding $x \in \mathbb{R}^d$ into a Cartesian product of sub-codebooks $\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_p$ for $\mathcal{S}_j \subseteq \mathbb{R}^m$, while retaining a complexity $\mathcal{O}(k \cdot d)$ for a lookup and space (see Figure 2.2c)

Keeping the same complexity while increasing the number of bins (using a larger p) enables a much finer granularity for the query process. Consequently, the vectors in each bin are much more coherent. The canonical projection is a mapping of each vector-part $[x]_p$ independently

$$s_p \colon \mathcal{X} \to \mathcal{S}_p, \quad x \mapsto s_p(x) := \arg\min_{s \in \mathcal{S}_p} \left\| [x]_p - s \right\|_2 , \tag{2.4}$$

to its nearest part-centroid $s \in \mathcal{S}_p$. The nearest centroid $c(x) \in \mathcal{C}$ for $x \in \mathbb{R}^d$ is the concatenation of the sub-centroids

$$c(x) = (s_1(x), s_2(x), \ldots, s_p(x))^\top . \tag{2.5}$$

Table 2.1: Creating enough bins using a small memory footprint is essential to ensure informative bins during the quantization. VQ and PQ use the same amount of memory but provide a different level of granularity.

	Number of Addressable Centroids			
k	$p = 1 (VQ)$	$p = 2$ (PQ)	$p = 4$ (PQ)	$p = 8$ (PQ)
4	4	16	256	65536
8	8	64	4096	16777216
16	16	256	65536	4294967296
32	32	1024	1048576	1099511627776

Finding a good quantizer $c(\cdot)$ for $\mathcal{X}$ can be formulated as finding p sub-codebooks $\mathcal{S}_1, \mathcal{S}_1, \ldots, \mathcal{S}_P$ independently, which can also be done using Lloyd iterations. Therefore, when setting $p = 1$, Product Quantization becomes Vector Quantization.

It is indeed easy to produce exponentially many (with respect to (w.r.t.) p) clusters using PQ as illustrated in Table 2.1. However, many will be empty as the intrinsic dimension $\bar{d}$ of the dataset can be smaller than, $d > \bar{d}$. Assuming the set of query vectors share a similar distribution with the set of database vectors (a common assumption), we can expect that most queries will also correspond to non-empty clusters. Nonetheless, we still must be able to deal with clusters of highly multifaceted cardinality as illustrated in Figure 2.2.

2.1.2 IVFADC and Extensions

The full approach Inverted File With Asymmetric Distance computation (IVFADC) of Jégou *et al.* [JDS11] combines both ideas of VQ and PQ when constructing an index structure for large-scale datasets. Hereby, the VQ acts as a coarse quantizer with k_1 bins $\mathcal{C}_{\text{coarse}}$ in a first stage and each coarse cluster $C_j \in \mathcal{C}_{\text{coarse}}$ contains its own PQ index structure applied to the residual $r = x - c_{\text{coarse}}(x)$. Retrieving a set of possible candidate vectors during a query is based on traversing both index structures.

Extensions For a better quantization, the authors of *optimized-PQ* (OPQ) [Ge+13] propose to augment the PQ index structure using the mapping

$$c\colon \mathcal{X} \to \mathcal{C}, \quad x \mapsto c(x) := \arg\min_{c \in \mathcal{C}} \delta(Rx, c), \tag{2.6}$$

where $R \in \mathbb{R}^{d \times d}$ is a $d \times d$ rotation matrix. Another extension *locally-optimized-PQ* (LOPQ) [KA14] uses a individual rotation matrix R_j for each coarse cluster

$$c_{\text{coarse},1}, c_{\text{coarse},2}, \ldots, c_{\text{coarse},j} \ldots, c_{\text{coarse},k_1}$$

and quantizes the residual of a rotated query vector version $R_j q - c(q)$.

While at first glance, it seems to be appealing to rotate the index structure resp. the dataset globally [Ge+13] or even locally [KA14], it requires a high-dimensional matrix multiplication slowing down the entire query time — even when based on a GPU implementation. As multiple matrix multiplications require a severe computational effort, the authors of LOPQ [KA14] precompute all possible projections during the offline phase. However, this approach is only practical when query vectors are known beforehand.

2.1.3 Inverted Multi-Index (IMI)

The inverted multi-index [BL12] exploits PQ rather than VQ for an indexing structure over all database vectors, which reduces the number of centroid-distance calculation for cluster proposals or vise-versa increases the number of bins: Given part distances to k codebook vectors, for each part $[q]_p$ of the query vector q this approach sorts the corresponding k centroids w.r.t. to the ascending distances

$$\begin{array}{lllllll} [q]_1 \rightarrow & \{i_{11}, & i_{12}, & i_{13}, & \ldots, & i_{1k}\} & = \mathcal{I}_1 \\ [q]_2 \rightarrow & \{i_{21}, & i_{22}, & i_{23}, & \ldots, & i_{2k}\} & = \mathcal{I}_2 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ [q]_p \rightarrow & \{i_{p1}, & i_{p2}, & i_{p3}, & \ldots, & i_{pk}\} & = \mathcal{I}_p, \end{array}$$

where i_{23} is the index of the 3rd nearest cluster for part $p = 2$. The combined cluster indices of all parts encoded a bin index via a multi-index

$$i \in \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_p. \tag{2.7}$$

For a query, starting with bin B_i for $i = (i_{11}, i_{21}, \ldots, i_{p1})$ a heuristic is needed to traverse all bins B_i in the vicinity. Babenko *et al.* [BL12] make use of a priority queue to dynamically select the next closest not yet visited bin until sufficiently many bins are proposed. All vectors stored in each visited bin B_i are then examined in an exhaustive search using PQ-based re-ranking of the residual to each bin centroid.

2.1.4 Spotting the Computational Bottleneck

Both methods IVFADC [JDS11] and IMI [BL12] achieve state-of-the-art precision but have efficiency issues when making a query. VQ-based indexing requires a vast number of full-vector comparisons of high-dimensionality codebook vectors, and even for PQ-based indexing the number is still substantial.

A typical search space is SIFT1B (see Table 2.2), which contains 1 billion SIFT vectors. When doing a query for $q \in \mathbb{R}^d$, the IVFADC approach [JDS11] uses $k_1 = 8192$ clusters in the initial coarse VQ quantizer to extract possible subsets

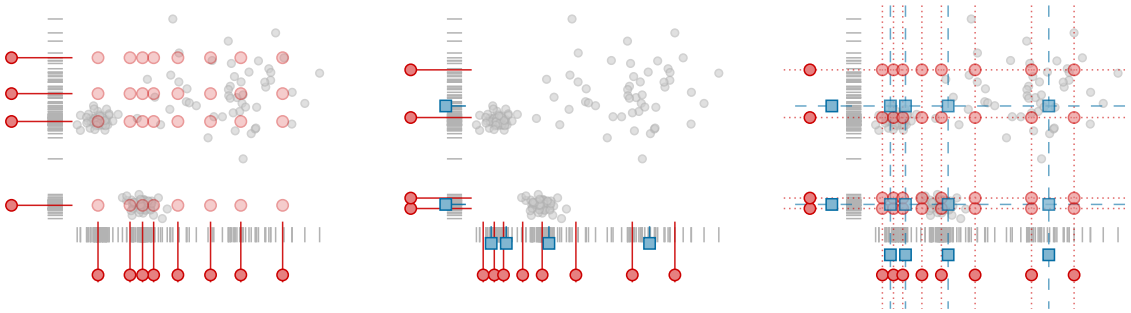
Table 2.2: Standard datasets used for benchmarking ANN approaches containing ground-truth data, when performing queries from a query set disjoint from the data set.

Dataset	Dimension (d)	Base Set (n)	Query Set(n_q)
DEEP1M	256	10^6	10^3
SIFT1M	128	10^6	10^4
GIST1M	960	10^6	10^3
SIFT1B (BigANN)	128	10^9	10^4

in the search space. The number of candidates is empirically set to 0.05% of the database size to find the nearest neighbor with probabilty ≥ 0.9 by visiting 64 clusters [JDS11]. This corresponds to 524288 candidate vectors for each query in the SIFT1B database. Thus, this approach requires k exact d-dimensional distance calculations for each query vector q to identify reasonable clusters. The centroids are then sorted based on distances, and the w-best clusters are chosen, giving a coarse list of database vectors $\mathcal{L}_c \subset \mathcal{X}$ which has a high chance of containing the nearest neighbor. These vectors are again sorted in a re-ranking step based on PQ of the expensive residual-computation $r_w = q - c_w$ to the identified cluster c_w, which are precomputed [JDS11], [KA14] and stored in a distance lookup-table. Again, this precomputation is only possible when query vectors are known beforehand. The distance between the query vector $q \in \mathbb{R}^d$ and each nearest neighbor candidate $x \in \mathcal{L}_c$ can be approximated by quantizing the residual using a second PQ codebook with k_2 words. Re-sorting the list using a better approximation from PQ gives a *filtered* list $\mathcal{L}_c \to \mathcal{L}_f$. Considering only the first few vectors $\mathcal{L}'_f \subset \mathcal{L}_f$, an exhaustive search in $\mathcal{L}'_f$ becomes feasible.

The lookup and re-ranking steps when visiting w clusters require $k_1 + w \cdot k_2$ exact distance calculations during query time. With typical values of $k_1 = 8192, w = 64, k_2 = 256$ (compare [JDS11]), this implies 24576 full distance calculations – excluding possible matrix multiplications [Ge+13; KA14].

To put these values into context, we benchmarked a GPU-based approach using an NVIDIA Titan X. for computing $2^{14} = 16384$ exact distances under optimal conditions, which takes 0.13ms. This provides a lower bound when transferring the previously described methods in a practical setting to a hypothetical ideal GPU implementation omitting latency and other overheads. In the next section, we introduce a hierarchical approach reducing the total number of exact distance calculations to less than 200. As computing 256 exact distance computations takes 0.0021 ms under ideal conditions, the number of exact vector comparison represent a bottleneck. A *complete* query in our algorithm, which we will describe now, only takes about 0.02 ms in total due to a combination of an efficient hierarchical index structure and the parallel nature of our approach.



a) PQ b) hierarchical clustering c) PQT

Figure 2.3: Three different quantization schemes with $k = 32$ clusters. Vector Quantization (a) represents vectors by their closest centroids. Product Quantization performs the clustering in subspaces (here axes) (b). A tree structure can be used to build a hierarchy of clusters on each axis (c). Our method use the hierarchy of two quantization levels, first using PQ with a low number of centroids, and then a second-layer of PQ within these bins (d). Points drawn as ▫ are PQ centroids, and each corresponding cluster is split again into finer 4 clusters (2 on each axis) with centroids illustrated as ● .

2.2 PQ-Tree (PQT)

Previous proposed methods [BL12; JDS11; KA14] require quantizing the residual within each bin for re-ranking, which is accelerated by precomputing these values. However, with unknown query vectors, such optimization cannot be made. Additionally, these methods are hindered by a slow enumeration of the next best bin [BL12].

We address these issue by reducing the number of exact vector comparisons based on a PQT. Our approach presents an efficient heuristic for proposing bins, as well as a novel re-ranking method based on projections to quantized lines for re-ranking. Our re-ranking step is particularly efficient as it can merely reuse distance calculations computed during the tree traversal. Finally, we demonstrate that our approach can be efficiently implemented on a GPU.

The main idea is that product quantization is performed using a hierarchical VQ-tree [ML14] for each part rather than a flat codebook. The tree structure on the centroids speeds up the query (*online*), sorting into the database (*offline*), and indexes considerably more bins in contrast to the inverted multi-index. Additionally, it is designed to enable the reuse of computed values for fast re-ranking.

2.2.1 Tree Structure - Offline Phase

Sorting each database vector $x \in \mathcal{X}$ into a bin B_ℓ gives disjoint sets, $\mathcal{X} = \dot{\bigcup}_{\ell=1}^{k} B_\ell$. We describe how to effectively map a vector into a bin, $m \colon \mathcal{X} \to \mathcal{I}_1 \times \mathcal{I}_2 \times \cdots \times \mathcal{I}_p$.

The indexing structure is a tree which consists of two levels of quantizers. The first level is a traditional p-parts product quantizer with k_1 centroids for each part. Each resulting part cluster is then *independently* refined by one additional vector quantizer with k_2 centroids as illustrated in Figure 2.4. The bins are addressed by any combination of the per-part child node centroids. Mapping a database vector $x \in \mathcal{X}$ to one of $n = (k_1 \cdot k_2)^p$ bins allows us to prune the entire search space by only picking bins during a query which are likely to contain the nearest neighbor.

The same figure shows a PQT trained on SIFT1M and projected to 2D using random coordinates. Thereby, each pixel represents a point p. The gray-value is defined as $0.8f_1 + 0.2f_2$, where f_k is the ratio between the smallest two distances from p to a centroid from the k-th layer in the PQT to emphasize the hierarchy. To find nearest neighbor candidates for a query ● the PQT prunes the search space to the highlighted area ($w = 1$), which itself is then further clustered.

Training the codebook. Constructing the VQ-trees is done independently for each part, first by constructing a VQ codebook (level 1) using Lloyd iteration in the fashion of the Linde-Buzo-Gray algorithm [LBG80] and then quantizing all sub-vectors (level 2) assigned to a first level cluster.

While the inverted multi-index approach [BL12] also uses two levels of product quantization, the second is exclusively used for re-ranking. As opposed to this, we use two levels for indexing. While additional tree levels are possible, we empirically found this configuration to be optimal in terms of balancing the number of bins to check with the reduction of candidate vectors.

2.2.2 Query - Online Phase

A query now consists of four steps: tree traversal, bin proposal, vector proposal, and re-ranking. The tree traversal is carried out as described above producing an ordered list of $(i, d)_p^2$ for the best subset of level 2 clusters.

Tree Traversal. The tree reduces the number of exact distance computations required during traversal by pruning. After comparing to all k_1 first-level codebook vectors, the distances are sorted, and only the w best clusters are refined for further distance calculations for the level 2 codebook. Let $y \in \mathbb{R}^d$ be the query vector, distances $\delta([y]_p, [c^1]_p)$ to the k_1 first-level separately computed for each part. This step returns a set of IDs and distance pairs

$$\{(i, \delta')_p^1 \mid \delta' = \delta([y]_p, \left[c_i^1\right]_p)\} \tag{2.8}$$

for each part p and each level-1 centroid c_j^1.

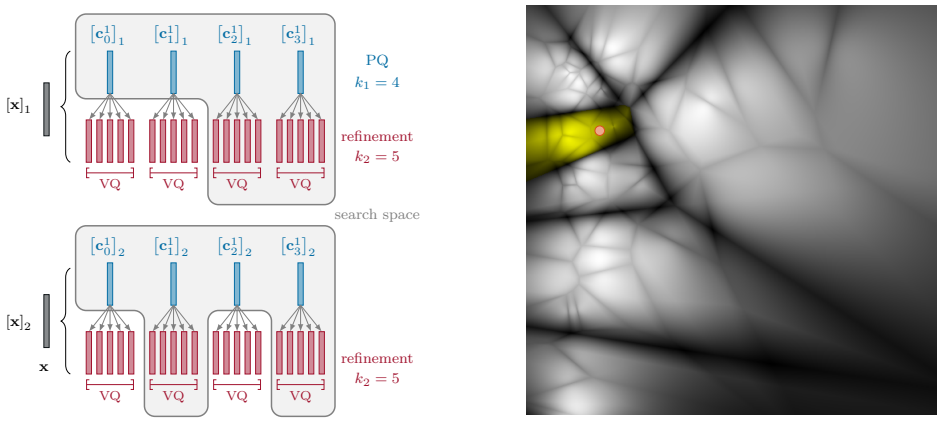


Figure 2.4: Left: Both parts of $([x]_1, [x]_2) \in \mathbb{R}^d$ are quantized by a VQ tree with $k_1 = 4$ clusters in the first and $k_2 = 5$ finer clusters in the second level. During traversal, only the best w closest clusters of the first level are refined. The example search space by extending $w = 2$ clusters is illustrated as the gray area. Right: Voronoi-cells from first and second level clusters of the index-structure obtained from the SIFT1M experiment when projected onto two random coordinates. The highlighted search region corresponds to the best first level clusters for the illustrated query.

From these possible per-part clusters, we only use the closest w centroids for further processing, *i.e.* computing the distances $\delta([y]_p, [c^2]_p)$ only to those level 2 centroids whose corresponding c^1 are in the best set. The level-2 distances are ordered to find the best cluster indices for each part. Finally, combining the best indices of the individual parts identifies the best bin as in Equation 2.7.

A typical configuration might consist of four parts ($p = 4, k_1 = 16, k_2 = 16, w = 4$), amounting to only $16 + 4 \cdot 16 = 80$ full vector distance calculations to address $(16 \cdot 16)^4 \approx 4$ trillion bins. For practical purposes, we applied modulo-hashing by using unsigned integers representing the index.

Bin Proposal Heuristic. Given the best bin as determined by the index $i = (i_{11}, i_{21}, \ldots, i_{p1})$. Due to the quantization nature of this approach, this bin might not contain the actual nearest neighbor point. Instead, one has to find a sequence of neighbor bins to ensure that a sufficient number of vectors for re-ranking is generated. The priority queue used in Babenko and Lemptsky [BL12] would yield the optimal sequence but it requires a resorting operation for each proposed bin, which is expensive and is sequential by nature. This operation is the green solution illustrated in Figure 2.5.

Instead, we propose to choose a precomputed fixed traversal heuristic. The most simple order would be to compute all id-vectors $v \in \{1, r\}^p$ and sort them according to their distance from the origin $\|v\|_2$. However, this returns an isotropic bin

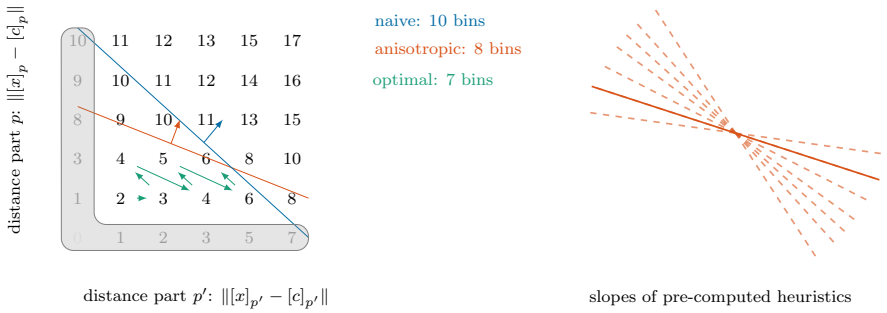


Figure 2.5: Merging the independent lookups from differents parts p, p' to find the best bin-combination requires sorting all combinations. A Dijkstra-based traversal [BL12] (green) cannot be evaluated on a GPU due to its sequential nature, though it is the optimal sequence. Illustrations of possible parallel approximations are a naive (blue) or an anisotropic (orange) heuristic.

traversal heuristic as depicted in Figure 2.5 (blue line) compared to the optimal sequence from [BL12] (green lines) and our proposed anisotropic traversal heuristic (red line). The anisotropic version with flexible slope produced a better approximation of the Dijkstra ordering. As the anisotropic version is a generalization, it already covers the isotropic version and therefore represents a better relaxation of the priority queue traversal. Hereby, we precompute bin orderings for 10 slopes 1.08^k with $k = -5, -4, \ldots, 4$. Each slope describes the progress balance on one part-pair. A slope of 1 would equally handle both parts, while a slope of 1.08^{-5} would allow more bin combinations with higher IDs in the second part (see orange lines in Figure 2.5).

2.2.3 Re-ranking by Line Quantization

In the index structure, each database vector is quantized to its nearest bin with a quantization error Δ_i. To find the best vectors in the bin, they need to be sorted based on their distance to the query vector. However, a full d-dimensional distance calculation for each vector is too expensive. Similarly, re-ranking based on product quantized residuals [JDS11] requires comparison to yet another codebook.

Inspired by the Johnson-Lindenstrauss lemma [IM98], we propose a novel line quantization, where some of the information gathered during traversal is reused.

Offline computation Each vector (●) is quantized to the nearest projection (●) onto any line (■-■) through the level 1 centroids for each part, see Figure 2.6. For multiple parts, this quantization effectively spans hyper-planes. The distance of the query point to the line quantized vector can be evaluated exactly and efficiently

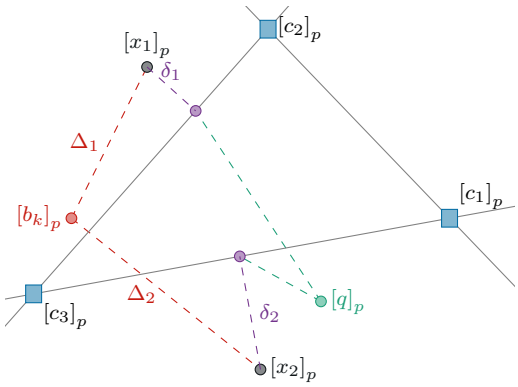


Figure 2.6: Line Quantization. In traditional PQ, each database vector part $[x_i]_p$ (●) is projected onto the bin centroid part $[b_k]_p$ (●) yielding an approximation error Δ_i. Vectors in a bin would be indistinguishable from each other given a query y. We project $[x_i]_p$ onto (●) on the nearest line ■-■ which gives an approximation error δ_i making vectors within a bin distinguishable. Computing δ_i can reuse intermediate values without the need of a full distance computation.

using only one 2D triangle calculation per part.

In order to disambiguate the database vectors x in these bins, we propose an approximation by projecting each vector part $[x]_p$ in the linear subspace

$$\text{span}([c_i]_p, [c_j]_p), c_i, c_j \in \mathcal{C} \tag{2.9}$$

defined by the first level centroids $[c_i]_p$ and $[c_j]_p$ illustrated by ■ in Figure 2.6. Thereby $[c_i]_p, [c_j]_p$ are chosen such that the quantization error δ_p is minimized. Therefore, the calculation of the distance $\delta(y, x), x \in \mathcal{X}$ is not an impediment for efficient lookups as each database vector part $[x]_p \approx (1 - \lambda_p)[c_i]_p + \lambda_p [c_j]_p$ can be approximated by drawing on existing information from the tree-structure.

Hence, it is sufficient to store λ_p and (i_p, j_p) for each database vector, where the information of λ_p and the indicies from $[c_i]_p, [c_j]_p$ in Figure 2.7 describes the approximation (●) of a vector (●). In fact, all information about a database vector $x_i \in \mathcal{X}$ we need for the complete algorithm is encoded in the $3 \cdot p$ tuple

$$x_i \leftrightarrow (\lambda_1, \ldots, \lambda_p, i_1, \ldots, i_p, j_1, \ldots, j_p), \tag{2.10}$$

which can be heavily compressed to 2 bytes per part p in our implementation. The value λ_k is quantized to fit `uint8_t` and $i_k \cdot j_k < 256$ in practise. While this scheme does not have the same compression rate as previous methods, it is the first to allow an efficient parallel re-ranking on the GPU by look-up from already computed values without any computational overhead. For $p \leq 4$ parts storing these values directly on the GPU is possible even for the SIFT1B dataset.

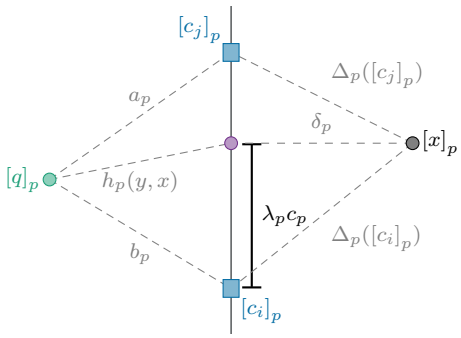


Figure 2.7: Exact query to line calculation. The database vector part ($[x]_p$, ●) is projected onto (●) at the line $([c_i]_p, [c_j]_p)$ with error δ_p. When re-ranking the exact distance h_p between the query ($[y]_p$ ●) and the quantized database point is obtained using triangulation. All necessary values are known as they are computed during tree traversal (a_p, b_p) or during database construction $(c_p, \lambda_p, (i, j))$.

Additionally, storing one global lookup table of $p \times k_1 \times k_1$ precomputed distances between all pairs of level 1 centroids, *i.e.* $\| [c_s]_p - [c_t]_p \|_2^2$ for all p, s, t is required. But this can be computed in the offline phase as it is independent of query vectors.

While the size of this lookup table looks huge at first glance, it contains at most 1024 entries in practice. Hence, using $p = 8$ forces $K = k_1 \cdot k_2 < 32$ to allow each vector ID fit into a 32bit integer. Therefore, the lookup table contains only 1024 entries for $k_1 = 4, k_2 = 8$ and $P = 8$. In this case, even for the SIFT1B dataset, at most each 4th bin is only filled[1].

Online computation During tree traversal all distances between a query point $y \in \mathbb{R}^D$ and all level 1 centroids have already been computed as list of pairs $(i, d)_p^1$. The approximate distance to the database vector x is computed given the triple (λ_p, i_p, j_p), looking up a_p and b_p in the query's list, and c_p. The distance between y and x is approximated by

$$d(y,x)^2 = \sum_{p=1}^{p} d([y]_p, [x]_p)^2 \approx \sum_{p=1}^{p} h_p(y,x)^2 \tag{2.11}$$

$$\approx \sum_{p=1}^{p} \left(b_p^2 + \lambda_p^2 \cdot c_p^2 + \lambda_p \cdot (a_p^2 - b_p^2 - c_p^2) \right). \tag{2.12}$$

Note, that it is possible to compute the distance between a query and database vector by triangulation *exactly* up to the projection error[2] δ_i as illustrated by the

[1] There are $(k_1 \cdot k_2)^p = (4 * 8)^8$ bins.

[2] The induced distortion does not impair the query process. Please refer to Section 2.3.3 for more details.

SIFT1M					
		Recall			
Method	Query Time (ms)	R@1	R@10	R@100	Speedup
FLANN [ML14]	5.32	0.97	n.a.	n.a.	×9.6
LOPQ [KA14]	51.1	0.51	0.93	0.97	×1
IVFADC [JDS11]	11.2	0.28	0.70	0.93	×4.5
Ours: PQT_1 (CPU)	4.89	0.45	0.86	0.98	×10.4
Ours: PQT_2 (CPU)	5.74	0.98	(exact re-ranking)		×8.9
Ours: PQT (GPU)	0.02	0.51	0.83	0.86	×2555
Ours: GPU brutef.	23.7	1	-	-	×2

GIST1M		
		Recall
Method	Query Time (ms)	R@100
FLANN [ML14]	n.a.	n.a.
LSH [Dat+04]	22.7	0.132
IVFADC [JDS11]	65.9	0.744
Ours: PQT(CPU)	63	0.83

Table 2.3: Performance on the SIFT1M (left) and GIST1M (right) dataset using different methods. Reported query times include query + re-ranking times. The GPU implementation uses the first 2^{12} vectors from the proposed bins and $(64 \cdot 8)^4$ bins. The reported CPU performance is base on $(8 \cdot 4)^2$ bins. PQT_2 is PQT_1 but with additional exact re-ranking.

dashed line in Figure 2.7.

In practice we use different numbers of parts for the tree ($P_{\text{tree}} = 2$ or 4) and for the line quantization ($P_{\text{line}} = 8, 16$ or 32) for sufficiently precise re-ranking. Using exactly the same level 1 codebook with p parts, we split each centroid part to get $p' = k \cdot p$ parts and compute the distances by aggregating the components.

2.3 Results

We now present the results of the PQT evaluated on several standard benchmark sets. All reported CPU query times were obtained from a single-threaded C++ implementation using SSE2 instructions. Results of our GPU implementations are obtained with a NVIDIA GTX Titan X.

We use the publicly available benchmark SIFT1M, SIFT1B datasets [Jég+11], of 128-dim vectors and GIST1M [JDS11] of 960-dim vectors. For the codebook training process, we solely use the first 100K/1M vectors from the respective datasets. It was not possible to obtain any results on GIST1M using FLANN in Table 2.3.

2.3.1 Query Times and Recall

We compared our implementation with the available implementations [KA14; BL12]. Due to the approximation nature of these algorithms and discrete parameter space, it is non-trivial to find parameter settings which produce the same accuracy for timing comparisons. Therefore, we choose a highly optimized GPU-based exhaustive search as a strong baseline method. The accuracy is measured in *recall* (Recall at k (R@k)), which is the fraction of nearest neighbors found in the first k proposed vectors after re-ranking.

Table 2.3 gives the average query time in milliseconds obtained on the same machine using publicly available code. Compared to all PQ-based approaches [BL12; KA14] our approach ($P_{\text{line}} = 32$) is faster on the CPU at similar accuracy. Allowing [KA14] to use more memory consumption for re-ranking slows down the query process. Note that the reported time of Kalantidis *et al.* [KA14] excludes all intensive operations like the multiplication of query vector with a $d \times d$ rotation matrix, which were precomputed.

On the GPU, sorting the SIFT1M vectors into the bins takes 1051ms, performing the line quantization for these 1M vectors about 458ms ($p = 4, k_1 = 16, k_2 = 8, w = 8$). The processing time for one query is roughly 39 microseconds, split into 4% traversal, 35% bin selection, 11% vector proposal, and 50% re-ranking. In our implementation, the maximum number of sortable vectors on the GPU per query is currently limited to 4096 during re-ranking. Applying different algorithms, this restriction could be removed.

With the right configuration of bins, high recall values can even be achieved on the SIFT1B data set (Figure 2.8). Because the full data set did not fit on the

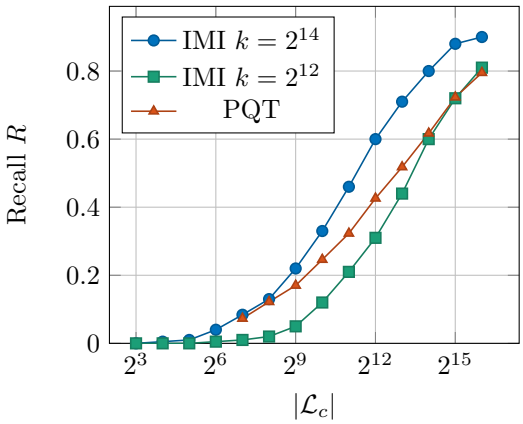


Figure 2.8: Recall rates on the SIFT1B data set ($p = 4, k_1 = 32, k_2 = 16, w = 8$) with ordering of bins. The recall from PQT is without a reranking step. Even with significant lower query time, our approach is comparable in quality to the inverted multi-index (IMI) with $k = 2^{12}$.

GPU, the database was build in waves of 1M vectors, aggregating the information on the CPU. With file I/O this offline phase took about 144min. On an NVIDIA GTX Titan X with 12GB of RAM one can upload the resulting DB structure, i.e., bin sizes and vector IDs per bin. For the SIFT1B dataset, it was essential to resort the proposed bins by the actual distance. This slowed down query time to 0.027ms in total without re-ranking. The recall rate (R@k) at 10 of our approach is $R@10 = 0.35$. For the re-ranking, we directly accessed the CPU main memory from the GPU resulting in a total query time of 0.15ms.

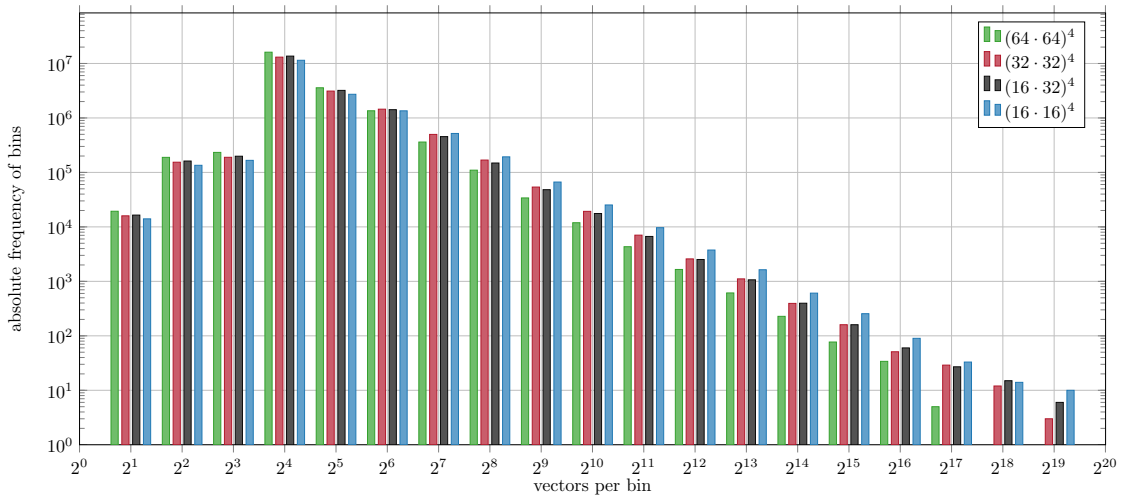


Figure 2.9: The SIFT1B dataset is split up into $\max\{10^8, (k_1 \cdot k_2)^4\}$ bins by a PQT with hashing. Each bar above 2^ℓ counts the number of bins which contains $[2^{\ell-1}, 2^\ell]$ vectors. More cluster centers yield fewer highly occupied bins. Distributions with smaller median and shorter tail are better.

Memory is the limiting factor for the maximum number of actual bins. We apply hashing to 100M bins. Increasing the number of parts P or introducing a further level into the tree would further boost the number of bins – at the same time, also the number of bins to be visited in the vicinity would drastically increase and slow down the system.

2.3.2 Distribution of Vectors

While mapping a database vector to a bin reduces the entire search space, those bins may contain millions of vectors when the total number of bins b is small. On the other hand, traversing million of bins is infeasible, even on the GPU. Figure 2.9 contains the bin-histogram for the SIFT1B dataset (Table 2.4) in combination with hashing for a maximum of 100M bins. Using higher values of b allows a much finer granularity of produced bins.

By restricting the number of bins to be at most 100M by hashing, bins are unions of several different clusters. A re-ranking step would only pick vectors from the correct cluster, because of the smaller approximate distance.

2.3.3 Precision of Line Quantization

The line projection approach for re-ranking proposed nearest neighbor candidates uses a lossy compression of the original datapoints. When choosing the correct value of P_{line} usually the tradeoff between better approximation (higher values of

	Absolute Frequency of Bins			
Vectors per Bin	$(64 \cdot 64)^4$	$(32 \cdot 32)^4$	$(32 \cdot 16)^4$	$(16 \cdot 16)^4$
2	19,381	15,963	16,473	14,044
4	189,062	154,028	161,860	135,162
8	233,141	189,513	198,391	166,573
16	16,122,413	13,076,618	13,654,802	11,451,256
32	3,581,330	3,115,776	3,206,650	2,720,408
64	1,354,592	1,449,036	1,418,639	1,351,564
128	361,308	498,769	454,400	519,172
256	110,007	168,674	148,859	193,192
512	34,012	53,860	48,198	66,601
1,024	11,925	19,287	17,610	25,269
2,048	4,327	7,056	6,678	9,670
4,096	1,654	2,585	2,521	3,761
8,192	613	1,111	1,067	1,634
16,384	228	394	397	608
32,768	77	160	160	255
65,536	34	51	60	90
131,072	5	29	27	33
262,144	0	12	15	14
524,288	0	3	6	10
1,048,576	0	0	0	8
2,097,152	0	0	0	0

Table 2.4: Numerical values of Figure 2.9.

P_{line}) and less memory consumption (smaller values of P_{line}) arises. To illustrated the effect of our re-ranking approach with $P_{\text{line}} \ll d$, we applied PQT with re-ranking to the MNIST dataset of handwritten-digests $\mathcal{X} \subseteq \mathbb{R}^{784}$ using $P_{\text{line}} = 28$ (see Figure 2.10), where $\mathcal{L}_c$ comes from the bin traversal is re-ranked using line quantization.

We tested the performance of encoding each database vector $x \in \mathcal{X}$ by its projection onto a line for different numbers of parts used for line quantization (see Figure 2.11). The recall rate increases with the number of line parts P_{line}. Low quantization errors with moderate computational and storage effort are obtained with $P_{\text{line}} = 16$. The necessary data for each query vector is hereby directly assembled during the tree traversal without the need for any further quantization computation.

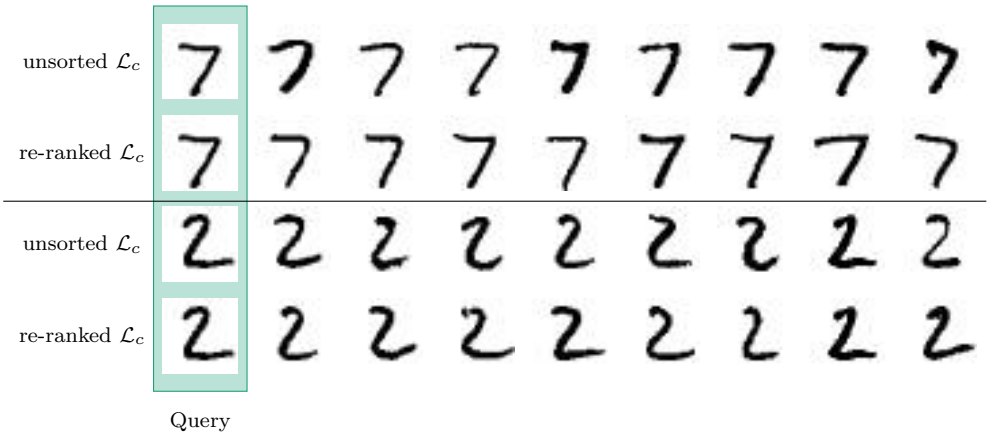
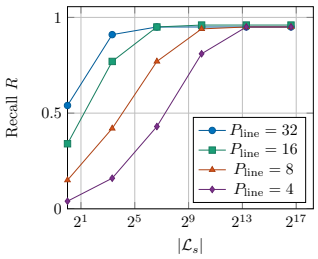


Figure 2.10: List of nearest neighbor candidates $\mathcal{L}_s$ (resp. $\mathcal{L}_c$) on MNIST for a query from the test set. Re-ranking these $d = 784$ dimensional raw image vectors was done with our line projection method ($P_{\text{line}} = 28$).



P_{line}	min distor.	max distor.	avg. distor.	time (ms)
2	10874.9	179870	30534.6	2.0
4	8967.8	166722	26257.9	2.6
8	6709.2	145082	19719.4	3.6
16	3318.3	84640	10509.7	5.3
32	1035.3	39143	3686.71	8.5

Figure 2.11: Line Quantization recall on SIFT1M using $p = 2, k_1 = 16, k_2 = 8, w = 4$ (single-threaded CPU, $|\mathcal{L}_c| < 20000$).

2.4 Possible Future Work

There are several aspects which could lead to potential improvements. Given that all distances to first level clusters are known, the natural question arises if more sophisticated plane-quantization or cube-quantization schemes instead of the proposed line-quantization method might lead to better recall rates due to better approximations. A better approximation scheme requiring fewer parts P_{line} will counteract the overhead of storing and reading additional information. Identifying the trade-off and the best approximation approach would yield a more efficient traversal within bins and finally emerge in a more meaningful candidate list $\mathcal{L}_c$.

Currently, the ordering of the input dimension is fixed and therefore the parts only receive the same dimensions. Along with the observations that a perfectly sorted candidate list of the database does not change, when shuffling the coordinate axes

$x(x_1, x_2, \ldots, x_d) \mapsto \tilde{x}(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(d)})$ for a permutation[3] $\pi \in \mathrm{Sym}_d$. When re-ordering the coordinate axis, different parts will lead to different clusters. Using n re-orderings $\pi_1, \pi_2, \ldots, \pi_n$ in conjunction with n different index structures increases the recall accuracy, while pay the price of losing speed by factor n. Let $X_1, \ldots, X_n$ be random variables denoting the position of the correct nearest neighbor within the candidate list $\mathcal{L}_s$ from n different runs — assuming $(X_i)_{i=1,2,\ldots,n}$ iid. Hence, $F(m) := P(X_i \leq m)$ describes the probability of the nearest neighbor is found within the first m retrieves elements from the candidate list. It is easy to verify, that $P(\min(X_1, X_2, \ldots, X_n) \leq m) = 1 - (1 - F(m))^n$. Figure 2.12 plots the recall rates (R@k) from Figure 2.11 and expected improvements when shuffling the coordinates axis $n = 1, 2, \ldots, 6$ times. The (expected) recall rate R@10 increases from 77% to 95% for $P_{\text{line}} = 16$ when doubling the query time which eventually surpasses the accuracy from $P_{\text{line}} = 32$ (recall 91%).

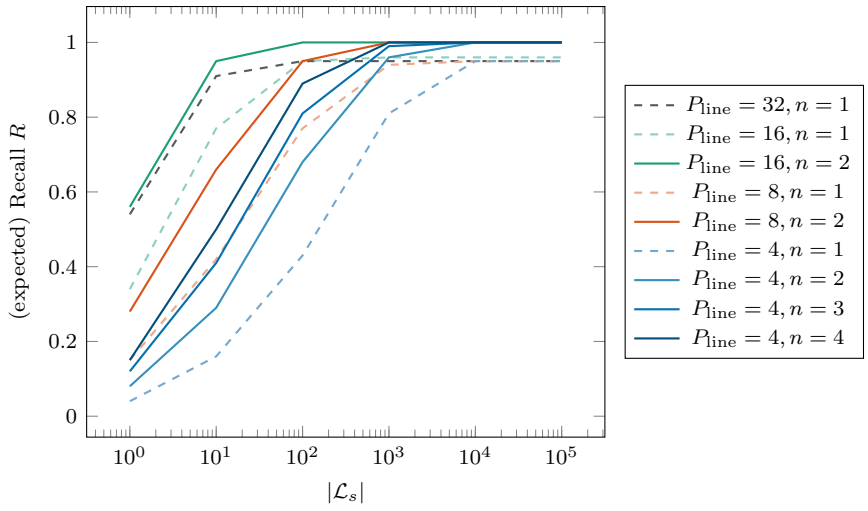


Figure 2.12: Expected improvements when accumulating the results of multiple runs $n \geq 1$ with different orderings of the coordinate axes.

Traversing the PQT to identify the first promising bin containing the nearest neighbor has minor computational costs compared to iterate neighboring bins in case of missing the correct nearest neighbor in the first bins. Currently, PQT traverses multiple paths (multiple bins) to increase the chance of finding the nearest neighbor, represents the main bottleneck in the algorithm — despite the anisotropic heuristic. Interconnecting bins vectors with their nearest neighbors vectors potentially across different bins would help to parallelize the query.

[3] Sym_d contains all bijections from the set of d symbols to itself.

Rule #28:
Check your gradients at least twice!
If they are wrong, neural networks will
ignore them and still learn something.

Chapter 3

Deep Learning: Fundamentals And Practice

Deep learning as a subfield of machine learning is the backbone for many of today's applications in computer vision. Machine learning methods regularly outshines other techniques when it comes to learning from observations. Thereby, predictions are made without a set of fixed symbolic rules and hand-crafted features. Deep learning methods usually refers to deep neural networks that have been trained to detect a pattern in a given training data set.

In this thesis we use deep neural networks for solving real-world problems. For the sake of completeness, we introduce relevant terminology and fundamental concepts to explain background material which eases the understanding of the following chapters. The experienced reader may thus skip to the next chapter. For a more in-depth introduction in learning from data, we recommend the Machine Learning Book [Mit97].

3.1 Introduction - Learning from Data

The famous painter Michelangelo left his painting "The Entombment" as an unfinished piece. Large areas of the canvas seems to be never be touched by any brush stroke of Michelangelo himself. While famous artists certainly can leverage several ways to capture a scene on a plain white canvas, they cannot escape their unique signature and technique – a pattern – during creating their art pieces.

We discuss how a deep neural network could hallucinate a finished version of "The Entombment" several hundred years later. Such a deep neural network can predict how the painting might look like, where even the smallest area is filled with details. Such a deep neural network hereby describes a special type of a function

$$f : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y} \subset \mathbb{R}^m, \tag{3.1}$$

which resembles a real relation $f_{\text{real}} : \mathcal{X} \to \mathcal{Y}$ between observed input data $x \in \mathcal{X}$ and expected output values $y \in \mathcal{Y}$. In the case of completing a painting, the

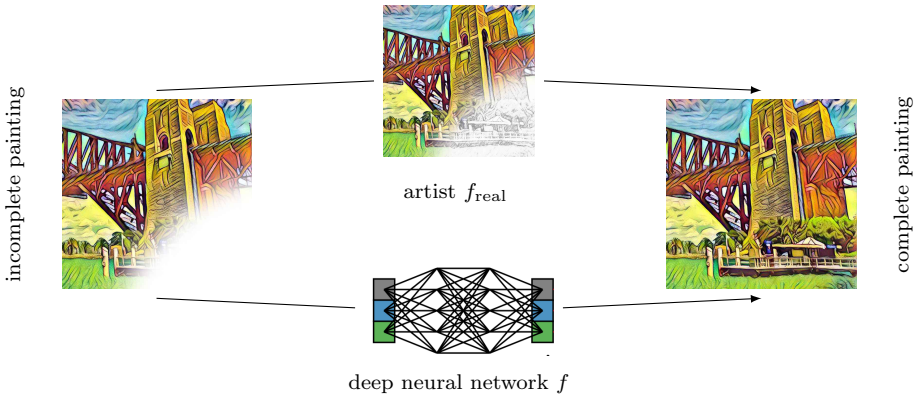


Figure 3.1: While a human artist would sketch missing parts and fill fine details in a sequence, a deep neural network can be trained to imitate this process and hallucinate the missing content in a single forward pass.

real relation f_{real} is the process of the artist delivering a finished painting. It is a mapping from an unfinished piece $x \in \mathcal{X}$ to an finished artwork $y \in \mathcal{Y}$ that we can inspect in today's galleries. The function can be modeled as

$$f_{\text{real}} : \mathcal{X} \subset \mathbb{R}^{h \times w \times 3} \to \mathcal{Y} \subset \mathbb{R}^{h \times w \times 3}. \tag{3.2}$$

It takes an RGB image $x \in \mathbb{R}^{h \times w \times 3}$ of size $h \times w$ with missing image parts and hallucinates content forming another image $y \in \mathbb{R}^{h \times w \times 3}$, where these parts are filled. This input representation is by no means unique. Instead of choosing a pixel representation, each stroke [ZJH19] could be encoded in a vector b_i with a certain brush type, color, pressure, direction and velocity. The function f_{real} could then predict the next stroke $f_{\text{real}}(b_i) = b_{i+1}$ forming a sequence of brush strokes. Further, the process of painting a picture can be describe in higher concepts of content (*e.g.* landscape, portrait) and painting genre (*e.g.* minimalism, impressionism). A function might map these high-level characteristics to a final painting.

Hence, the input data representation has a crucial impact on performance and interpretability of deep learning systems. Completing an RGB image representation in one pass is fast. But a brush-stroke based representation could illustrate the entire process of developing a painting.

Technically, f_{real} is approximated by the non-linear function f_Ω. And f_Ω is hereby parametrized by p tunable parameters $\Omega \in \mathbb{R}^p$ that can be freely chosen. Each single parameter in a highly non-linear function can dramatically change the behavior and output of the function f_Ω. In todays deep neural networks p typically ranges in the millions (*e.g.* 25.6 or 60 million parameters in ResNet-50 resp. ResNet-152 [He+16a]).

Training a deep neural network refers to the process of finding specific parameter *values* in Ω, that reduce the discrepancy $\ell\left(f_\Omega(x), f_{\text{real}}(x)\right)$ on some input data

$x \in \mathcal{X}$. The function ℓ describing the discrepancy is termed as *loss-function*. In the example of image in-painting the loss between f_Ω and f_{real} can be modeled as an Euclidean distance between $f_\Omega(x)$ and $f_{\text{real}}(x)$ in RGB space for a given image x. If the final appearance of an infinitesimal small area in a painting has only a few possibilities, the network can learn to classify these using the cross-entropy loss. Once the parameters Ω are optimized, a well-trained neural network f_Ω behaves similar to the real relation f_{real} and can predict $f_{\text{real}}(x)$ for any valid input x during the inference phase.

In summary, a deep learning approach draws on choices made in the representation of the data, the form of f_Ω and the way the loss is measured and minimized. While each part demonstrates a high degree of flexibility, the effectiveness of the overall approach relies heavily on the choices made.

3.2 Generating Training Data

As f_{real} is generally unknown, directly deriving the parameters Ω of f_Ω from f_{real} is infeasible. In some cases, sampling from f_{real} is costly and involves human annotations to generate pairs $(x_k, f_{\text{real}}(x_k))$ (*e.g.* image classification). When completing artworks, a recorded video of the painting process can deliver training data for the sequence of brush strokes. In some cases, fortunately, the inverse relation f_{real}^{-1} is well-understood and easier to compute than f_{real}. Digitally removing regions from finished paintings (f_{real}^{-1}) would produce input data x_k. Once enough observation pairs

$$\mathcal{T} = \{(x_1, f_{\text{real}}(x_1)), (x_2, f_{\text{real}}(x_2)), \ldots, (x_k, f_{\text{real}}(x_k))\} \tag{3.3}$$

are available as a training set $\mathcal{T}$, a deep neural network f_Ω can be optimized such that $f_\Omega(x_i) \approx f_{\text{real}}(x_i)$ holds for all k gathered training observations, *i.e.* $\ell\left(f_\Omega(x_i), f_{\text{real}}(x_i)\right)$ is minimal in Ω for all $i = 1, \ldots, k$.

3.2.1 Generalization Beyond Training Data

Ideally, f_Ω would be identical to f_{real} under all circumstance, *i.e.* the parameters Ω in f_Ω are chosen such that the expected risk[1] is minimal. This means that over *all possible* input-output pairs — which might not be part of data $\mathcal{T}$ — f_Ω and f_{real} behave similar. In such a situation a deep neural networks has learned to perfectly mimicking the way an artist fills an empty canvas. As the optimization of f_Ω can only be based on the finite amount of *observed* data, the expected risk cannot be directly minimized. Instead, the optimization procedure performs an empirical risk minimization on the observed data with bounded guarantees for the

[1] The expectation of the loss function.

expected risk [Vap92]. Presumably, a not surprising consequence of more observed data is a better match between the theoretical bound of the expected risk and the empirical risk that can be minimized in practice. In other words, incorporating more independent observed training examples (more paintings) significantly improves the generalization of f_Ω (a better chance to accurately mimicking yet unobserved samples of the painter's work). However, the expressiveness of f_Ω (*e.g.* number of parameters) counteracts this generalization guarantee [Vap92]. Therefore, the preparation of data comprises the technical challenge to store and handle enough data – besides solving the ambiguity of data representation.

Assuming f_{real} to be surjective, we might want to generate training data via sampling from

$$\left\{(f_{\text{real}}^{-1}(y), y) \,|\, y \in \mathcal{Y}\right\}. \tag{3.4}$$

Sampling training data according Equation (3.4) is preferable, when f_{real}^{-1} is easier to compute than f_{real}. While in the in-painting example from the beginning of this chapter the function f_{real}^{-1} can be directly evaluated (vanish information in a small patch), in some cases like deblurring videos in Chapter 6 or removing reflections in Chapter 5 neither f_{real} nor f_{real}^{-1} are easily accessible. But faithfully approximating f_{real}^{-1} by a computational less expensive but adequate function f_Ω^{-1} would effectively generate training data

$$\tilde{\mathcal{T}} = \left\{(f_\Omega^{-1}(y), y) \,|\, y \in \mathcal{Y}\right\} \tag{3.5}$$

on the fly. In Chapter 6 the approximation f_Ω^{-1} will replicate common causes of blur in videos and in Chapter 5 the physical process of reflection on surfaces under polarized light is faithfully modeled to synthesize training data.

While sampling training data from Equation (3.5) might produce plenty of data, the generation can be still costly compared to an training step of the deep neural network performed on the GPU.

3.2.2 Data Pipeline and Speed

Therefore, synthesizing training examples on-the-fly poses a trade-off between the number of possible observations for training and speed of generating these observations pairs. For example, the on-the-fly generation of blurry images (see Chapter 6) – while expensive – gives the benefit of independently sampling a specific blur for training and eventually increasing the number of independently synthesized blurry observation for training. A way to speed up this process is to store several patches of an blurry image and its sharp counterpart in advance on disk that can be digested later by the neural network at the cost of being limited by available memory.

This is the first decision to be made: What information should be stored on disk. Although each training example can be generated, fed through the network and disbanded, storing the data on the disk has the benefits of reproducible optimization

runs. This is especially useful during testing and comparing different network architectures and hyper-parameter choices. When it comes to the data storage format, several options exist. Training examples from the ImageNet Challenge [Den+09] are distributed directly as images[2]. While this eases the way of sharing, iterating all files can represent a severe bottleneck. To understand all common factors, parts of the system have to be benchmarked independently.

The combination of 8 consumer GPUs can process[3] 1536 images per second (1008 MB/second) during training, see Table 3.1. As the setup is based on PCI-e v2.0 connections between GPU memory and host memory, it has a maximal throughput of 8 GB/second. In this case the computation is clearly bounded by GPU computation. The only way to improve the speed would be using a smaller network architecture, *e.g.* ResNet-18 can process 1094 images/second on a single GPU.

Table 3.1: Raw speed (images per second) of ResNet-50 [He+16c]. Some configurations could not be tested due to missing hardware.

	Images/second (different number of GPUs)							
GPU - Type	1	2	3	4	5	6	7	8
Titan Xp	241	448	658	893	-	-	-	-
GeForce GTX 1080 Ti	213	396	587	791	841	1021	1200	1536
GeForce RTX 2080 Ti	243	537	-	-	-	-	-	-
Titan RTX	262	576	-	-	-	-	-	-

The data input pipeline must ensure that at least as many images are delivered as the bandwidth is capable of handling. This prevents the training process from being blocked by additional waiting times on the GPU side. Reading from an SSD (on the machine that has been used for the projects in this thesis) gives a speed of 460 MB/second with disabled cache and 9947 MB/second with cache enabled. Note, loading a single file into the cache has performance benefits if it is kept in the cache. A typical first optimization epoch (one entire optimization pass showing each image the deep neural network exactly once) is slow as the data is not cached yet. Throughout all following projects, each dataset has been stored as a Lightning Memory-Mapped Database (LMDB) file. Each entire subset (training/validation/test) is stored with all meta-data in a different LMDB file to guarantee performance and reproducibility. Once this file is loaded, the only remaining operation to switch between training examples is seeking within this file. An alternative to LMDB is the Hierarchical Data Format 5 (HDF5) and tf-Records

[2]Even the official dataset contains invalid images. An excellent first step is always to check the readability of the training data.

[3]Experiments were done in TensorFlow (v1.11, CUDA 10, CuDNN 7.3.0, Ubuntu 18.04) using the *tf.StagingArea* to keep the data on the GPU removing any non-GPU related overhead.

format, where the latter is part of the TensorFlow framework [Aba+15] and does not support random access. An experimental setup reading images in a random order from LMDB gives on average 333.84 MB/second compared to HDF5 with 173.52 MB/second in a single-threaded environment.

The standard strategy of handling training data in the following chapters is to load the data from the LMDB files in several independent processes. These independent processes augment and send the data over sockets using Zero-Em-Queue (ZMQ) to the primary training process which manages the (multi-)GPU interaction and consumes all the training examples. Separate processes are used for data reading and augmentation in Python as opposed to multi-threading. The default Python interpreter is slowed down by the Global-Interpreter-Lock. While this gives thread-safe memory management, a real multi-threading with parallel execution is therefore not possible. In Chapter 6, the Python implementation defers the computationally expensive part to separate CUDA/cuDNN implementations to blur an image.

Optimizing Neural Networks

Given a training set $\mathcal{T} = \{(x_i, y_i)_{i=1,2,\dots,n}\}$ of observations, we might hypothesize linear relationship between $x_j \in \mathbb{R}^d$ and $y_j \in \mathbb{R}^{d'}$ for the sake of simplicity:

$$f_{\text{real}}(x_j) =: y_j \approx f(x_j) = Wx_j, \quad W \in \mathbb{R}^{d' \times d}. \tag{3.6}$$

Again, the discrepancy between the model prediction (actual output, $f(x_j)$) and ground-truth (expected output, $f_{\text{real}}(x_j)$) can be measured. For the sake of simplicity we use the squared Euclidean loss function

$$\frac{1}{2n} \sum_{i=1}^{n} \|f(x_i) - f_{\text{real}}(x_i)\|_2^2 . \tag{3.7}$$

The optimal *weights* $W^\star$ yielding minimal error and can be obtained via solving the optimization problem

$$W^\star = \arg\min_W \sum_{i=1}^{n} \ell(f(x_i), y_i), \quad \ell(f(x_i), y_i) = \|Wx_i - y_i\|_2^2 . \tag{3.8}$$

Since, ℓ is a convex function [NW06, p. 8] in Equation (3.8), it is easy to verify that its unique minimum is obtained at $YX^\top(XX^\top)^{-1}$ when columns of X, Y represent the independent data points[4]. Typically the number of datapoints n in combination

[4]Usually, one uses the convention of per-row data points, which would complicate all following equations.

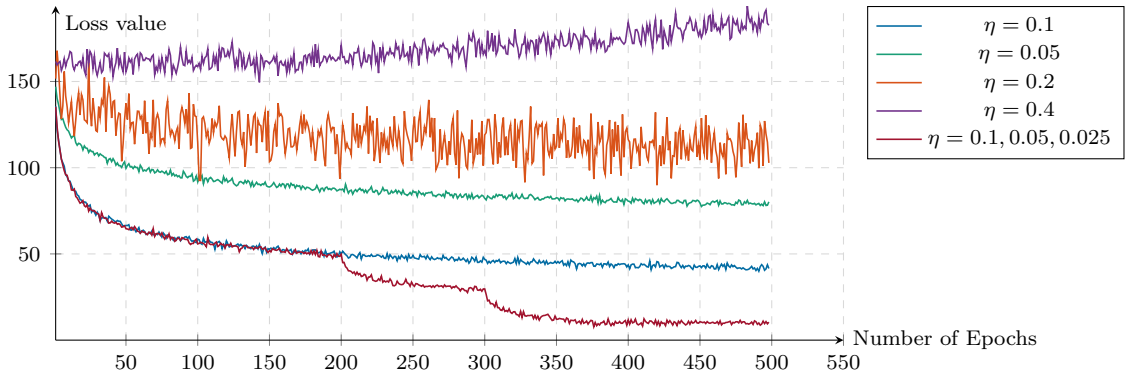


Figure 3.2: Behavior of loss function for different initial learning rates using the ADAM optimizer [KB14a] for experiments done with an earlier version of the network presented in Chapter 5. Even in the context of learning rate heuristics halving the learning rate when the error plateaus is a good strategy.

with high dimensions d, d' makes it infeasible to compute this solution directly in practice besides numerical issues when computing the pseudo-inverse $(XX^\top)^{-1}$ with a high condition number. Using an iterative approach like stochastic gradient descent [RM85; KW+52], the initial random guess W_0 is updated in the steepest-descent direction [NW06, p. 21] given by $-\nabla_W \ell(f(x_i), y_i)$:

$$W_{t+1} = W_t - \eta \cdot \nabla_W \ell(f(x_i), y_i) \tag{3.9}$$

$$= W_t - \eta \cdot \frac{\partial}{\partial W_t} \|W_t x_i - y_i\|_2^2 \tag{3.10}$$

$$= W_t - \eta \cdot 2(W_t \cdot x_i - y_i) \cdot x_i^\top \tag{3.11}$$

in a loop over each data point i by a pre-defined step-size η. In the convex setting the optimal step-size can be determined by the strong Wolfe condition [MT94; HZ06] to guarantee a decrease in the objective function. Unfortunately, the assumption of a linear and convex relation rarely holds in real-world problems. Deep convolutional neural networks then need to model a more complex relation. With this, η is treated as a hyperparameter choice and is usually adapted via some modern heuristics, *e.g.* ADAM [KB14a]. The specific choice of all optimizer's hyperparameters have an enormous impact on the fully trained model. To test the robustness of the proposed solutions, we rely on the defaults and choose an appropriate initial learning rate in the following chapters only.

Finding a good learning rate A helpful strategy to determine a good initial learning rate in practice is to first find a learning rate that will minimize the loss function in the first few iterations. To find a better learning rate value, we suggest

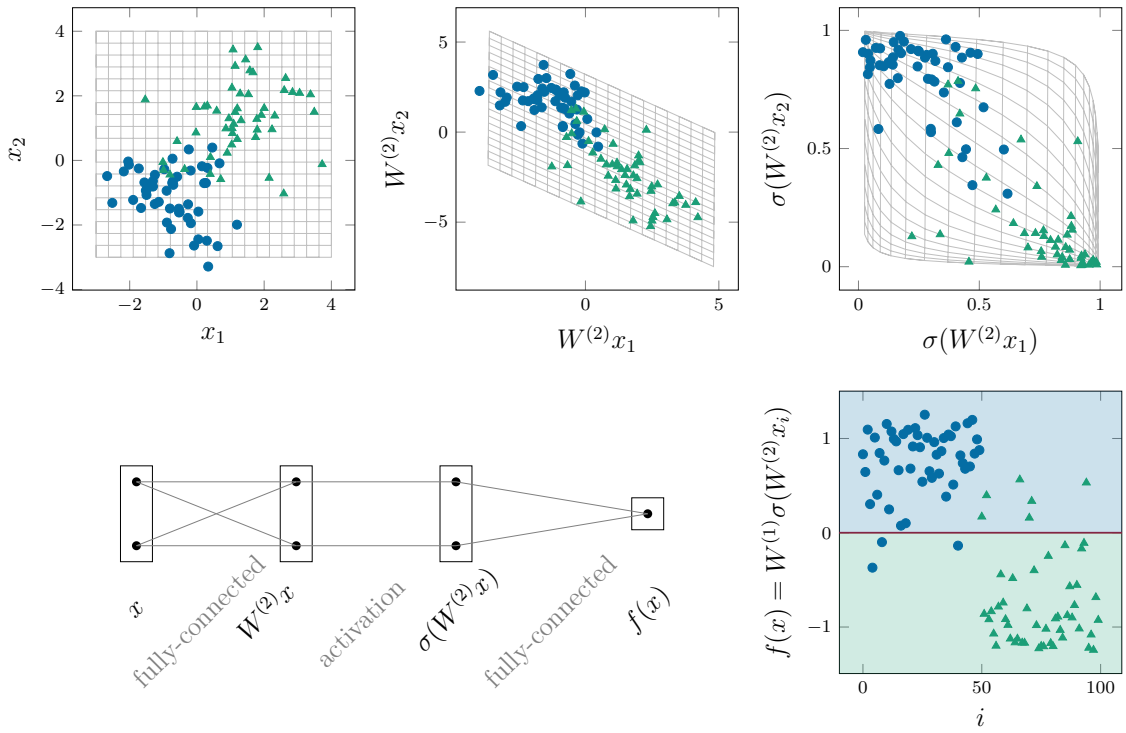


Figure 3.3: A deep neural network is trained to classify input data (top-left) from its position (x_1, x_2). The network consists of two fully-connected layers with parameters $W^{(2)}, W^{(1)}$ and a sigmoid activation function. For illustration purposes the training has been stopped after 20 iterations without final convergence.

to test several other choices of η by gradually increasing the value of the learning rate and run a few updates on an untrained neural network. This should be done until the learning rate leads to divergence within in the first few updates. Then the half of the highest learning rate, which still leads to convergent behavior, is usually a good choice. If the loss curve plateaus one can half the learning rate again. This process is illustrated in Figure 3.2.

To increase the model capabilities, a composition of basic functions is used, *e.g.*

$$f = f^{(1)} \circ f^{(2)} \circ f^{(3)}. \tag{3.12}$$

As the composition of multiple linear functions is linear itself, non-linear functions are interposed, *e.g.*

$$f(x_j) = W^{(1)} \underbrace{\left(\sigma\left(W^{(2)}x_j\right)\right)}_{\tilde{x_j}} = W^{(1)}\tilde{x_j}, \tag{3.13}$$

$$f^{(1)}(x) = W^{(1)}x, \quad f^{(2)}(x) = \sigma(x), \quad f^{(3)}(x) = W^{(2)}x. \tag{3.14}$$

Table 3.2: Description of a two-layer neural network for non-linear regression.

Common Term	Forward Pass	Backward Signal Multiplier
Squared Euclidean loss	$\|x-y\|_2^2$	c_a
Fully-connected Layer	$W^{(1)}x$	c_b
Activation Function	$\sigma(x)$	c_c
Fully-connected Layer	$W^{(2)}x$	$\partial \hat{x}_j / \partial W_t^{(2)}$

Hereby, σ is a non-linear function (typical choices are Sigmoid or ReLU [NH10]). This neural network is illustrated in Figure 3.3. The updates for the weights $W^{(1)}$ during the optimization procedure can be obtain via basic vector calculus

$$W_{t+1}^{(1)} = W_t^{(1)} - \eta \cdot \frac{\partial}{\partial W_t^{(1)}} \left\| W^{(1)} \tilde{x_j} - y_j \right\|_2^2 \tag{3.15}$$

$$= W_t^{(1)} - \eta \cdot 2(W_t^{(1)} \cdot \tilde{x_j} - y_i) \cdot \tilde{x_j}^\top. \tag{3.16}$$

The derivative w.r.t. $W_t^{(2)}$ is computed using chain-rule with automatic differentiation in reverse mode[5] (coined as "back-propagation"):

$$\frac{\partial}{\partial W_t^{(2)}} \left\| W_t^{(1)} \tilde{x_j} - y_i \right\|_2^2 \tag{3.17}$$

$$= \left[\frac{\partial}{\partial W_t^{(1)} \tilde{x_j}} \left\| W_t^{(1)} \tilde{x_j} - y_i \right\|_2^2 \right] \left[\frac{\partial W_t^{(1)} \tilde{x_j}}{\partial \tilde{x_j}} \right] \left[\frac{\partial \tilde{x_j}}{\partial W_t^{(2)}} \right] \tag{3.18}$$

$$= \left[2 \left(W_t^{(1)} \tilde{x_j} - y_i \right) \right] \left[\frac{\partial W_t^{(1)} \tilde{x_j}}{\partial \tilde{x_j}} \right] \left[\frac{\partial \sigma \left(W_t^{(2)} x_j \right)}{\partial W_t^{(2)}} \right] \tag{3.19}$$

$$= \left[W_t^{(1)} \right]^\top \left[2 \left(W_t^{(1)} \tilde{x_j} - y_i \right) \right] \left[\frac{\partial \sigma \left(W_t^{(2)} x_j \right)}{\partial W_t^{(2)}} \right] \tag{3.20}$$

$$= \underbrace{\left[W_t^{(1)} \right]^\top}_{c_b} \underbrace{\left[2 \left(W_t^{(1)} \tilde{x_j} - y_i \right) \right]}_{c_a} \underbrace{\left[\left(1 - \sigma \left(W_t^{(2)} x_j \right) \right) \sigma \left(W_t^{(2)} x_j \right) \right]}_{c_c} \frac{\partial W_t^{(2)} x_j}{\partial W_t^{(2)}} \tag{3.21}$$

$$= \underbrace{\left[W_t^{(1)} \right]^\top}_{c_b} \underbrace{\left[2 \left(W_t^{(1)} \tilde{x_j} - y_i \right) \right]}_{c_a} \underbrace{\left[\left(1 - \sigma \left(W_t^{(2)} x_j \right) \right) \sigma \left(W_t^{(2)} x_j \right) \right]}_{c_c} x_j^\top. \tag{3.22}$$

[5] Interestingly, there is no clear consensus of who invented back-propagation. A first publication about the derivation from the chain-rule is from Dreyfus [Dre62] in 1962 while the first practical implementation is described in the Master's thesis of Linnainmaa [Lin70] (1970). The first publication with application to neural networks is from Werbos [Wer81] (1981) and became popular after a paper by Rumelhart, Hinton and Williams in 1986 [RHW88].

Note, the scaling factors c_a, c_b can be computed independently from $W_t^{(2)}$. Each such building block is coined as a layer in a deep neural network, see Table 3.2. Hence, using the chain-rule for derivatives we can embed any novel layer in the network as long as the derivative of the layer output w.r.t. to its inputs is explicitly given. For example, it is possible to embed a Fisher-Vector representation [WGL17] as a neural network layer, convolution operation over a set of arbitrary points [GWL18] or a PatchMatch operation (extension of [Bar+09]).

A simple implementation of the derived update formulas can be done directly using NumPy. The entire training – without the claim to be an highly efficient implementation – of a two-layer neural network can be implemented as:

```
import numpy as np

x = np.concatenate([np.random.randn(2, 50) - 1,
                    np.random.randn(2, 50) + 1], axis=1)
y = np.concatenate([np.ones((1, 50)), -np.ones((1, 50))], axis=1)

W2 = np.random.randn(2,2)
W1 = np.random.randn(1,2)

eta = 0.01
for i in range(20):
    x_bar = W2.dot(x)
    x_tilde = sigmoid(x_bar)
    y_hat = W1.dot(x_tilde)
    loss = np.linalg.norm(y_hat - y)

    ca = 2 * (W1.dot(x_tilde) - y)
    cb = W1.T
    cc = (1- sigmoid(x_bar))*sigmoid(x_bar)

    grad_x_tilde  = cb.dot(ca)
    grad_x_bar = grad_x_tilde * cc
    gradW1 = 2 * (W1.dot(x_tilde) - y).dot(x_tilde.T)
    gradW2 = grad_x_bar.dot(x.T)

    W1 -= eta * gradW1
    W2 -= eta * gradW2
```

This implementation was used to generate the data for the illustration in Figure 3.3. Popular deep learning frameworks allows to define the forward pass as a graph. The framework then automatically constructs necessary operations for a given graph to run backpropagation without any user action. An implementation of the small neural network using TensorFlow [Aba+15] is:

```
import tensorflow as tf

x = tf.placeholder(dtype=tf.float32, [100, 2])
y = tf.placeholder(dtype=tf.float32, [100, ])

net = tf.layers.dense(x, 2, activation='sigmoid')
net = tf.layers.dense(net, 1)

loss = tf.reduce_mean(tf.squared_difference(net, y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)

with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  for i in range(20):
      sess.run(train, {x: ..., y: ...})
```

Recent deep neural networks rely on discrete 2D grid-based convolution operations (deep convolutional neural networks) to reduce the number of parameters which yields more efficient training. As a discrete 2D convolution be formulated as a matrix multiplication using a Toeplitz matrix all derived formulas from above can be used for these networks as well. Technically, most routines are indeed implemented on the GPU as a matrix multiplication [Che+14]. But the matrix multiplication scheme is adapted [CPS06] to unroll the convolution operations in a matrix multiplication and does not use a Toeplitz matrix. As todays neural networks mostly rely on convolutions with smaller kernel sizes more dedicated methods like Winograd [Win80] are even more effective.

Deep Network Architectures

Deep neural network architectures typically differ in the way f_Ω is modeled – especially how the layers interact. Since deep neural networks address non-convex problems, small changes in architecture can have a significant impact on the performance. One counter-intuitive observation is that deeper networks (although having larger capacity) yield more significant training errors [He+16a] without some adjustments. It has been empirically shown [He+16a] that using rather simple residual skip-connections $f(x) = h(x) + x$ (instead of $f(x) = h(x)$) eases the optimization procedure and improves performance. Hence, optimizing the weights towards zero in a residual skip-connection $h(x) = 0$ is therefore empirically easier than learning the identity mapping $f(x) = x$ when training deeper networks [He+16a].

Further, the choice of the loss-function highly depends on the underlying task. While image restoration approaches (Chapter 5 and 6) generally rely on the Euclidean loss in RGB space, learning an encoding of inputs in a high-dimensional

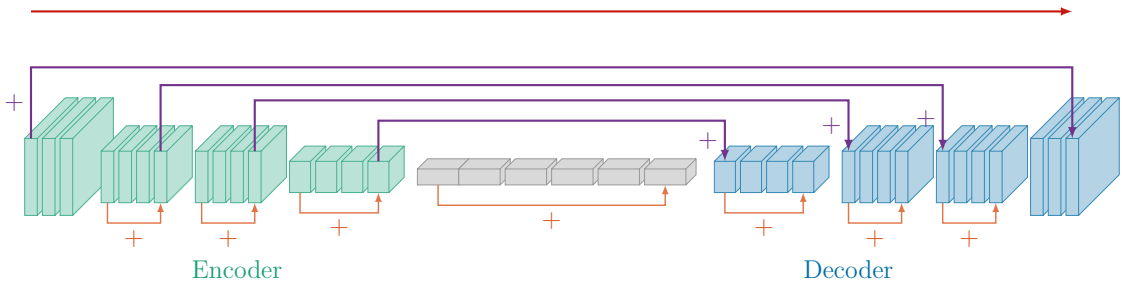


Figure 3.4: Illustration of an encoder-decoder network with ResNet [He+16a] (orange) and UNET skip-connections [RFB15] (green)

space (Chapter 4) involves a Triplet-Loss [HA14]. Still, exchanging the loss function changes the first backward signal multiplier c_a and has an impact on all subsequent back-propagation steps.

For image restoration task typically an encoder-decoder network structure is used, which can be written as

$$f = \underbrace{D_1 \circ D_2 \circ \cdots \circ D_{n-1} \circ D_n}_{\text{decoder } D} \circ \underbrace{E_n \circ E_{n-1} \circ \cdots \circ E_2 \circ E_1}_{\text{encoder } E}. \tag{3.23}$$

Hereby, E_i is usually a block of convolution layers followed by a down-sampling layer by either striding or some pooling operation (max-pooling, average pooling). Typically the decoder block D_i mirrors the encoder E_i block (see Figure 3.4) by assembling convolution layers and some up-sampling. Most networks architectures follow the suggestion of using nearest neighbor up-sampling [ODO16] to avoid checkerboard artifacts as a reverse operation to pooling.

One particular clever structure for encoder-decoder networks in image restoration task is the UNET [RFB15], see Figure 3.4. Instead of just having a neural network $\bar{f}$ learning a residual $\bar{f}(x) = x + f(x)$, where f_Ω is from Equation (3.23), all blocks have Unet-skip connections, *i.e.* a function h is applied to the output of the previous layer x and the corresponding encoder block result $E_i(x)$, *i.e.* $D_i(x) = h([x, E_i(x)])$.

3.3 Writing Custom Operations

In Chapters 4 and 6 custom TensorFlow operations have been used for speed considerations. Hence, part of the data generation pipeline was executed directly in the TensorFlow graph to benefit from local data availability on the GPU in combination with access to CUDA. The TensorFlow framework allows loading custom operation implementations living in a separate library dynamically. Each custom operation has a dedicated function to display the documentation in the Python front-end, a function to propagate input/output tensor description (data-type, shape). The

latter function is used to statically infer all tensor shapes throughout the entire network during the graph construction phase. The library further exposed functions encapsulated in a class that handles both phases: graph construction and execution. A typical custom operation is illustrated below:

```
class MyCustomOperation : public OpKernel {
 public:
  explicit MyCustomOperation(OpKernelConstruction* ctx) : OpKernel(ctx) {
    // Static checks during graph construction.
    // E.g., saving parameter choices.
  }

  void Compute(OpKernelContext* ctx) override {
    // Dynamic computations during graph execution.
    // Use all inputs 'ctx->input(i)' compute the final output tensor.
  }
};
```

While not officially supported[6] from the TensorFlow team, writing custom operation outside from the official source code repository is possible since TensorFlow v1.9. If a layer operation is differentiable, the backward pass as described in Section 3.2.2 is exposed as a separate custom operation independently from the forward pass. Registering the backward operation as a gradient to another operation is then done directly within Python.

[6] A documented way of compiling an example operation is given and tested in the project `https://github.com/PatWie/tensorflow-cmake` that has be maintained in parallel with the research projects in this thesis.

Rule #15:
Whenever a network does not converge, refine the idea not the random seed!

Chapter 4

Video Synchronization

Based on a un-annotated collection of videos with partly overlapping routes of car journeys, we present a tabula rasa learning approach, which is able to temporally align videos recorded months apart. Videos are synchronized, though they appear rather different due to weather and seasonal variations. The approach extends its training data set and autonomously selects training examples that serve as inputs for the training of a CNN.

The material of this chapter is based on the following publication:

[WFL17] Patrick Wieschollek, Ido Freeman, and Hendrik P. A. Lensch. „Learning Robust Video Synchronization without Annotations“. In: *IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2017. DOI: 10.1109/ICMLA.2017.0-173

While the previously described approach for approximate nearest neighbor search applies to all classical computer vision methods based on matching local features, understanding and assessing the current situation around us (humans) in a single glance is necessary to interact with the world as we know.

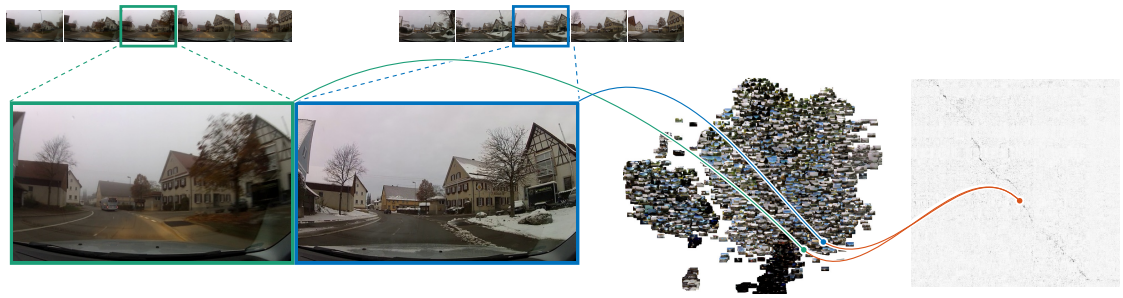


Figure 4.1: For any given frame our approach learns a global scene descriptor. Pairwise descriptor distances represent semantic similarity of the frames.

Looking at Figure 4.1 within a few tenths of a millisecond [GO09], humans can perceive and understand the scenes shown to conclude those highlighted frames indeed show the same scene although they are recorded at different times, with different illumination, motion blur, seasonal effects and a small spatial miss-alignment. For this task, we are focusing with ease on useful traits like the location of the houses or the shape of the road and mask irrelevant features such as traffic or the actual road conditions. From our experience we are familiar with all effects that might change the current appearance and we understand the global context. Given our speed in understanding the world around us, it is unlikely, that humans extract and match several scene-independent local descriptors, which then requires a nearest neighbor search as presented in the previous chapter (Chapter 2). It is more likely, that we will encode the entire scene from a glimpse using our global semantic understanding of the scene. Confronted with steady visual stimuli, we might learn such a scene abstraction from temporal information — we were never explicitly told which traits to look at to compare the two scenes in Figure 4.1.

This chapter focuses on training a convolutional neural network without providing such explicit information (annotations) for comparing multiple frames within the task of temporal video alignment (synchronization). Such a temporal video alignment is a dense frame-to-frame mapping between the involved videos in each time step with consistency along the time dimension. In contrast to classical approaches [Rüe+13; Aga+05; ST04; Wan+14] based on matching local descriptors, the presented neural network learns to describe an entire – potentially unseen – scene by a single descriptor. This method has two advantages over classical approaches: First, the network can learn to understand videos capturing the same content at different times, which might look completely different besides additional challenges such as ego- and object-motion or changes of the view angle and illumination. All these effects effectively hinder the extraction of meaningful local features in traditional computer vision methods when not relying on a global scene understanding. And second, a single learned descriptor per scene could avoid the computationally expensive step of matching multiple local features in a nearest neighbor search. This is essential, when processing hours of video content.

Although data-driven approaches like deep convolutional neural networks have proven excellent performance and capabilities in scene understanding [MMPN16], they usually require a large amount of high-quality labeled training data describing the underlying scenes accurately. One way to automate the labeling process for video synchronization would be to record synchronization signals such as Longitudinal Time Codes, genlock, GPS data or a landmark-based audio fingerprinting [BSM12] during acquisition. Solutions based on this kind of additional data are as accurate as the device which registered the data. Besides its limitation to out-door scenes, GPS has a typical precision of three meters [Gps]. While minor alignment errors would not be visible in vanilla image alignment, any non-frame accurate alignment would become apparent during simultaneous video playback.

Further, in many real-world scenarios, these explicit synchronization signals are not available as most consumer cameras only encode the creation date of the video file within the meta-data. Directly applying a learning based approach [Ara+16] to learn the alignment is not possible in this case. Producing a dense labeling by manual effort is not feasible either[1]. Our used video dataset consists of 28 million frames. Note, the ILSVRC challenge [Den+09] is based on 1.4 million labeled images only, which were annotated with by single label using crowdsourcing. It is worth mentioning, that in our setting, we deal with unstructured video content without any explicit knowledge about which frames or entire videos do match or not.

To overcome these problems, we propose a novel learning-based approach:

- Section 4.2 introduces a new challenging dataset for video-alignment covering rural scenes as well as city scenes across a year under different appearances.
- In Section 4.3 we propose a training protocol for training a neural network to match frames from different videos of the same scene without any annotation.
- Section 4.3.3 presents a method for robust identification and computation of matching tours for partially overlapping video pairs, which can automatically detect the start and end points of the matching tour.
- Section 4.4 evaluates the robustness and effectiveness of our algorithm, which will demonstrate significantly faster processing than competing methods and allows for robust synchronization of videos even under drastic appearance changes.

4.1 Related Work

The process of video alignment holds a natural relation to image alignment which was addressed by several studies [Bro92], *e.g.* using stereo correspondence estimation [SS02] or robust pixel descriptors [LYT11]. Algorithms like video stitching for creating panoramic videos [Aga+05], automatic summarization of videos [NMZ05], HDR video generation [Kan+03], vehicle detection for advanced driver assistance systems [Die+11] and video-copy detection [BBK10] among others are heavily dependent on such a robust and accurate temporal alignment of video-frames between multiple videos.

Basic video alignment is commonly used in the field of human action retrieval or surveillance motion capture. Here, finding similarities of human actions in videos are based on dynamic time warping of various sensor features to track the human skeleton [ZD09]. Bazin *et al.* introduced ActionSnapping [BSH16] which focuses on synchronizing actions performed by humans such as weightlifting, baseball pitching or dancing assuming a static background scene and frontal views.

[1] An attempt to thoroughly aligning a video pair of 8-minute length by hand took 41 minutes.

Most approaches for spatio-temporal video alignment [LY07; WZ06; TG04; CI02; UI06] assume a linear temporal correspondence, *i.e.* either a constant time shift between two videos or a constant change in playback speed for one video. Sand and Teller [ST04] compute a matching-likelihood of 3D motion to match videos. A non-linear solution was proposed by Wang *et al.* [Wan+14] based on a matching histogram of SIFT vectors using nearest neighbor search. Here, the search space dramatically increases with the length of the video sequence. In terms of application the most similar work to ours is from Evangelidis *et al.* [EB11; EB13], which allows for sub-frame accurate alignments of at most one-minute video snippets under negligible appearance changes on rather simple street scenes. Another related task is to recognize places from different view angles. A data-driven version of VLAD descriptors by Arandjelovic *et al.* [Ara+16] demonstrates the capability of neural networks to detect specific locations that are already present in the training dataset. Compared to a coarse place recognition, video alignment however requires a much finer temporal resolution. Figure 4.2 illustrates typical examples of frame-pairs from different datasets which are considered as similar for the specific task.

Learning similarities by training neural networks has been done previously for very specific applications such as signature verification [Bro+94], face recognition [CHL05; VHW16] and comparing image patches for depth estimation [ZK15]. These work rely on datasets with extensive human annotations and reliable ground-truth data.

4.2 Dataset

Our underlying dataset comprises of 602 full-HD, 30fps videos[2] (1.8 TB of raw data) capturing 260 hours of commuter's car journeys on partially overlapping routes between April 2012 and March 2013 and spanning over approximately 16,000km. The videos were captured using a GoPro Hero 2 camera mounted on the dashboard before every journey without any specific adjustments. However, the view angle does not feature differences as significant as in place recognition tasks [Ara+16].

The acquired videos feature both rural landscapes and urban scenes under varying traffic conditions such as temporary roadworks, rush hour, diverse weather conditions, *e.g.*, snowfall, rain, and seasonal environment appearance, for example, effects of vegetation as well as different daytime illumination (see Figure 4.2d).

Most videos show journeys between the same two cities but still have a variation in start and end locations and the actual roads driven. The temporal alignment of the videos thus poses a further challenge when trying to match stationary situations like waiting in traffic during rush hour with a video showing little traffic. This is different to linear video alignment tasks like copy-detection [Dou+16].

[2] frames per second (fps)

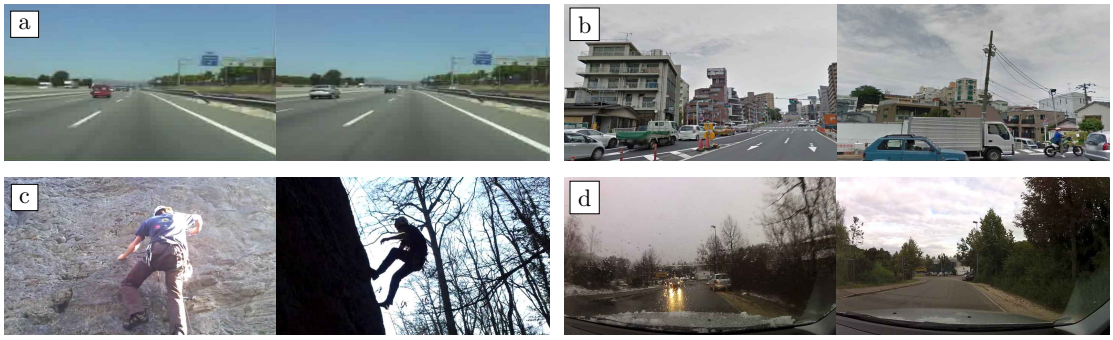


Figure 4.2: Comparing the datasets of related work. Each pair shows a typical matching frame-pair. (a) [EB11] only shows small changes in appearance and perspective, (b) [Tor+15] has a totally different view angles and (c) [Dou+16] requires only a constant time shift for alignment. For our dataset (d), an alignment method has to handle a different appearance and produce a playback with non-linear speed adjustment. (Images from the specified datasets.)

The enormous range of possible variations in the video content which also suffers from a noisy acquisition (*e.g.* different viewing angles, wipers, raindrops, camera transformation, etc.) requires an understanding of the entire scene context rather than a simple local feature matching like histogram-based methods [Wan+14; BSH16; EB13]. Moreover, the videos might show interrupted content, *e.g.* when the lens is cleaned or remounted during recording.

4.3 Method

We propose a framework to learn descriptors for all frames such that the Euclidean distance represents a similarity metric between captured scenes or locations. By properly designing the training protocol this generic approach features fast matching computation, robustness against seasonal effects and it does not rely on pre-existing labels for training.

The similarity metric can be exploited to robustly synchronize videos as will be explained in Section 4.3.3. While we establish the procedure to align partially overlapping video pairs (without knowing the exact frames) it is straightforward to extend it to multiple videos in a collection (see Figure 4.16).

Tabula-Rasa Learning. The algorithm alternates between two steps: the *learning* step and the *label generation* step. In the learning step, we assume given labels ℓ and train a neural network to produce meaningful descriptors for a similarity metric δ. The label-generation is based on the current version of the trained network

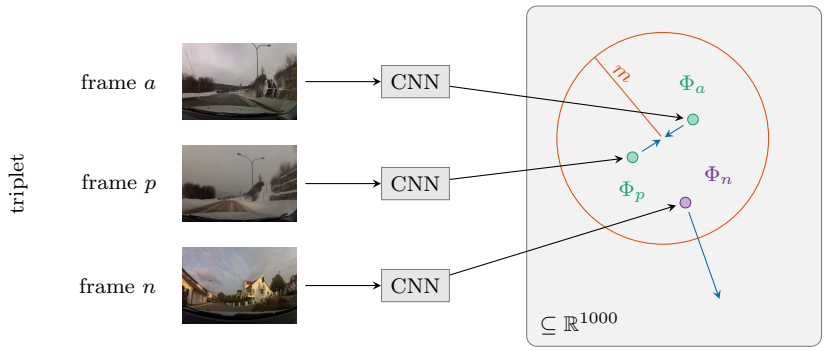


Figure 4.3: Let (a, p, n) be a frame-triplet with descriptors (Φ_a, Φ_p, Φ_n) produced by the same CNN. During training, the CNN is optimized to encode positive frames p close to the anchor point a, while negative frames n are pulled away from the anchor point if the distance is smaller than a margin m.

combined with tour matching (Section 4.3.3) to exploit temporal consistency. Given the learned similarity space, potentially more reliable labels are produced, replacing the old ones.

Over multiple iterations, more and more sophisticated and more informative training data is generated. None of these steps requires any human annotation nor recorded GPS signals.

4.3.1 Learning Step

In the learning step we want to learn an encoding of frames to establish a similarity measure between individual frames (x, y). Relying on the currently available labels from the training data we train a CNN [LeC+95] to predict a high-dimensional descriptor Φ_i for each frame i such that the Euclidean distance

$$(x, y) \mapsto \delta(x, y) := \|\Phi_x - \Phi_y\|_2 \tag{4.1}$$

is small when the frames are similar and vice versa. We use the standard ResNet-50 architecture [He+15] and add a projection from the *pool5* layer to learn the 1000-dimensional descriptors Φ_i. As SIFTs tend to learn edge filters in the first layers, we use a pre-trained ResNet version for object recognition as initialization.

In order to efficiently train the concept of similarity the triplet neural network approach [HA14] with weight-sharing is used, which generalizes well to unseen examples. It requires labels $\ell = (a, p, n)$ in the form of triplets of frames: for an anchor frame a the label needs one similar or positive frame p and one negative/dissimilar frame n. The similarity metric δ (Eq. (4.1)) is enforced by minimizing the triplet

loss by Hoffer *et al.* [HA14]

$$L(a,p,n) = \left[m + \|\Phi_a - \Phi_p\|_2^2 - \|\Phi_a - \Phi_n\|_2^2\right]_+ \tag{4.2}$$

for some margin $m \in \mathbb{R}$. The first term penalizes encodings of similar frame-pairs (a,p) that are too far away from each other in the high-dimensional feature space. The latter penalizes encodings of negative (non-matching) frame-pairs (a,n) if they are too close to each other (closer than some margin m – Figure 4.3). In practice, we constrain this encoding to live on the d-dimensional hypersphere, *i.e.* $\|\Phi_i\|_2 = 1$ and set m to 0.5.

Despite tagged by a high-dimensional descriptor, these learned encodings can be projected into 2d using t-SNE [MH08]. We sampled 20 frames per video from our entire video collection and placed these frame according to their encoding projections, see Figure 4.4. Clearly, all learned Φ_i are mostly independent of the actual appearance of a specific scene. Scenes showing the same content, *e.g.* rural scenes, downtown areas or forest scenes are clustered together. This is independent of the appearance, *e.g.*, rural scenes feature all conditions: dusk, fog, blue sky or sunsets. Only frames captured at night are packed into a cluster independent from their content.

4.3.2 Label Generating Step

While the learning step is rather straight-forward the challenge lies in automatically generating appropriate frame-triplet labels as training data. This label-generating step automatically harvests new training data for subsequent learning steps by explicitly exploiting the coherence in videos and by proposing and judging video alignments based on current encodings. The goal is to gather more and more informative training data in each iteration by successively increasing the complexity, *i.e.* to find positive frame pairs which show the same scene location but with a potentially different appearance as well as finding negative pairs which currently are assigned rather similar encodings. This is achieved in three waves:

Iteration 0:	*Intra-video* sampling of nearby frames for initial training.
Iteration 1+:	*Inter-video* sampling of frames from matching tours.
Iteration 2+:	*Transitive inter-video* sampling of frames from matching tours by propagation of alignments to other videos.

After each iteration of the label-generating step, which produces an augmentation of the training dataset, we re-train the neural network in the learning step, alternating between training and label generation.

Figure 4.4: Visualization of input frames from our database positioned by their learned encodings projected into 2d using t-SNE [MH08]. Rual scenes are located on the top left, downtown areas in the middle and forrest scenes on the right. Only, night drives are hard to encode properly due to rare information within the headlight beam.

Iteration 0. In Iteration 0 one has to solve the dilemma of generating reliable labels ℓ without having any trained network for proposing distances δ. Instead, we rely on the inherent coherence within the same video. Any arbitrary frame-pair which is at most 15 frames apart serves as positive sample (a, p). Any other random frame sufficiently far away from a is regarded as a negative frame n.

Iteration 1+. After the first iteration the network is trained on the given task and can produce features Φ_i for each frame i which is carried out for every 10th frame of all videos in the dataset. One can use the estimated Φ_i for approximating the pair-wise similarity, but since the network is not fully trained yet, it is essential to disambiguate between frame-pair encoding distances we can rely on or not.

The criteria for accepting a positive frame pair or detecting a negative one in this step is based on temporal coherence by potential matching tours between distinct videos (X, Y). A playback along this tour yields a synchronized video pair. A cost matrix

$$C = (\delta(\Phi_x, \Phi_y))_{x,y}, \quad x \in X, y \in Y \tag{4.3}$$

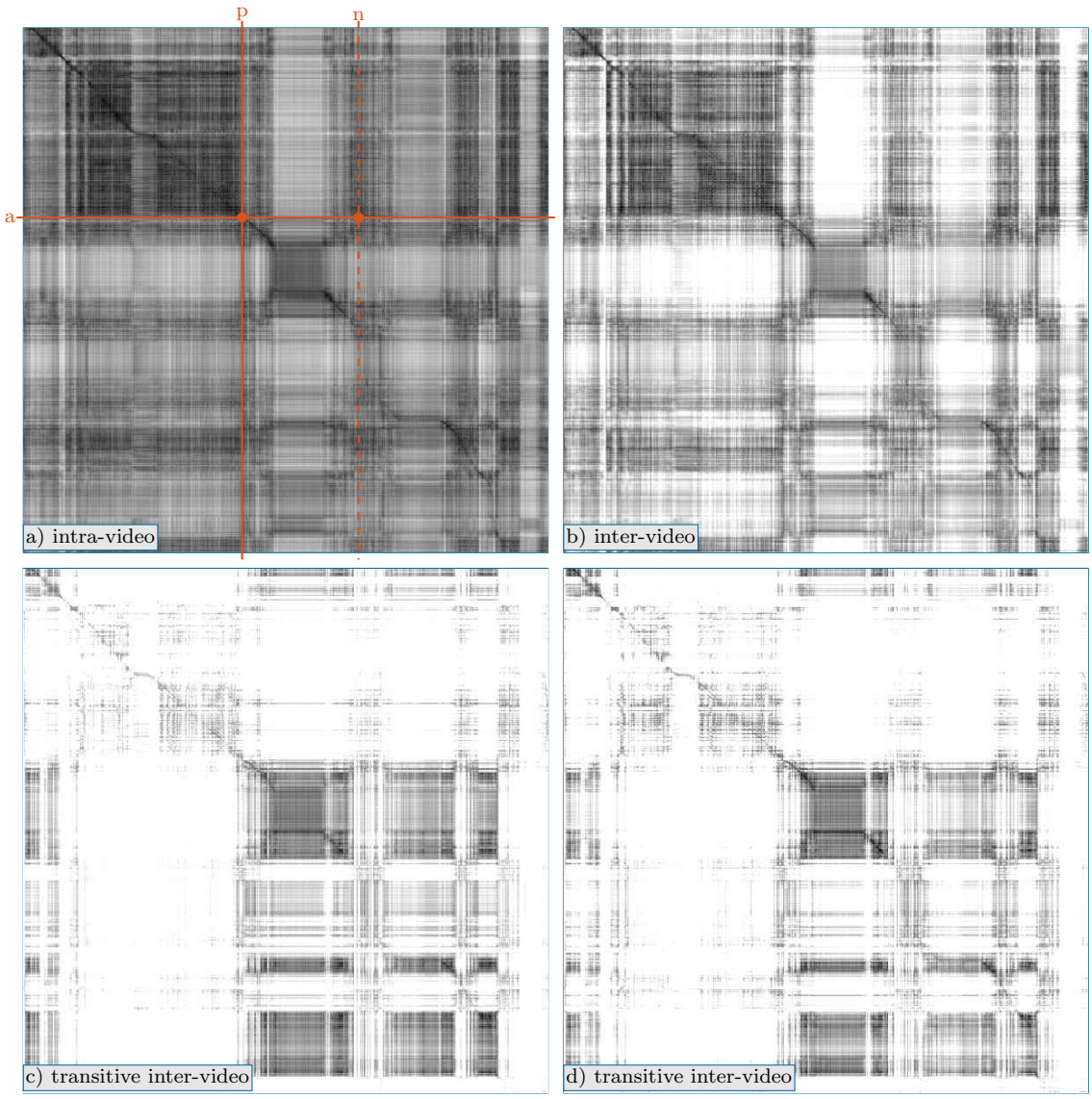


Figure 4.5: Development of the cost matrix of the same video pair throughout the iterative training process. Dark entries represent frame-pairs that are considered similar having a small distance in the feature space. Note how the path (red) of similar frame-pairs becomes more distinct. Large coherent regions along the path indicate stand-still, e.g., waiting at a traffic light. We further highlight matching frames a, p and hard negative n.

can be computed containing all coarse pair-wise distances of frames. Corresponding frames should have a small distance. Though this does not hold all cases in early iterations (since the network is not fully trained yet), it might already be possible to detect a matching tour as explained in Section 4.3.3, *i.e.* to find an optimal path through the cost matrix (see Figure 4.5). If the path is sufficiently distinct a playback of both videos along the path will synchronize them correctly and all nodes on the path resemble positive pairs with row a and column p, independent of the currently proposed measure $\delta(a, p)$. Some new positive pairs might be found this way giving hints on how to optimize the encoding in the next training phase. Similarly, we can harvest challenging negative pairs (a, n) by just choosing a column n sufficiently far away from the path. Most informative will be such a pair if $\delta(a, n)$ is rather small, indicating the frames clearly should be rated distinct but are not yet.

Though we might not find all possible paths between all videos yet, the resulting labels $\ell(a, p, n)$ will be more informative than in the previous iteration as the appearance between two videos will more likely be different even for matching frames. The process is visualized in Figure 4.5 where the correct matching path becomes more obvious and easier to detect throughout the iterative training. An improved encoding Φ_i, which results from the augmented training data, compare to intra-video sampling (Figure 4.5 a)) results in a cost matrix with a clearer path, see Figure 4.5 b). It is important to note that this new information does not come out of nowhere. Our heuristic of detecting false-positive (new hard negatives) and false-negatives (frames pairs on the path with relatively high costs) forms this new information.

Iteration 2+. Computing a (coarse) cost matrix for each single video pair is inefficient. When the network becomes more robust during training, we can propagate detected matching tours transitively to other pairs — without the need of computing all cost matrices. The property of "X and Y have a matching tour", denoted as $X \sim Y$, is an equivalence relation (meeting the requirements reflexivity, symmetry and transitivity). Therefore, we propose a transitive sampling. In fact, our entire dataset can be split into distinct equivalence classes[3] $V/\sim$ under the relation $X \sim V$ when videos X and V share a part of the same tour. Hence for any two videos X and Y from $V/\sim$ we already know the existence of a matching tour $X \sim V$ and $V \sim Y$. Transitivity also directly gives us a matching tour for $X \sim Y$ if both share some overlap. This allows us to sparsely sample video-pairs and to propagate matching frames across videos. Using a tree-based index structure reduces the complexity to $\mathcal{O}(n \log n)$ when synchronizing n videos (see Figure 4.16). Due to transitivity, one can establish matching tours which so far could not be

[3] A more figurative description is a clustering of all videos assigning them to an exemplary video with partly the same tour as a cluster center.

Figure 4.6: Extracted hard-negatives pairs, *i.e.* frame pairs (a, n) with a small distance $\delta(a, n)$ between their encodings Φ_a, Φ_n. Looking at these frame-pairs without temporal context it is hard to tell whether they belong to the same place or not.

detected using the previously trained encodings. With this iterative process we can quickly generate a huge number of challenging and informative training triplets $\ell(a, p, n)$ even for videos which have been captured months apart or the appearance difference makes them elusive for our heuristic in previous iterations. Starting in iteration two we equally mix the obtained training examples from inter-video sampling (Iteration 1+) and transitive inter-video sampling (Iteration 2+).

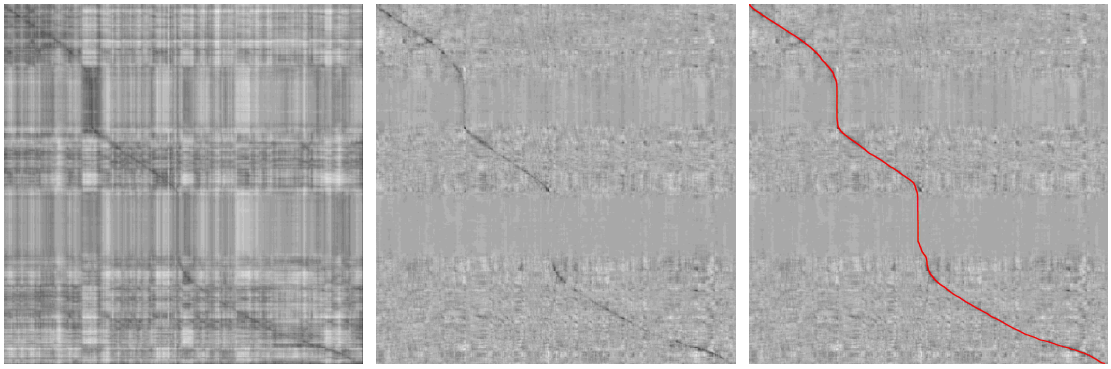
Training facilitates hard negative sample mining in all iterations (as described above) mixed with random negative with $p = 0.5$. Hard negatives are frames formerly adjudged as somewhat similar by the network and even humans are hard-pressed to examine whether they depict the same scene or not, see Figure 4.6.

4.3.3 Robust Tour Matching

We will now describe how to find a matching tour given a cost matrix C. The entire iterative scheme is based on robustly detecting false-positives resp. false-negatives from the network prediction and producing complex training data in a reliable way.

Pre-processing: De-correlate Costs

Particularly in the early iterations, the similarity matrices C produced by the CNN contain a lot of false predictions, since it is not yet fully adapted to the task. These errors exhibit a low-rank structure because any frame that is not correctly encoded is likely to corrupt an entire column (or row) of the similarity matrix (left of Figure 4.7). Additionally, some of the pairs are, indeed rather similar although we would like to treat them as different. For example, many journeys through rural areas with little information but crop fields on both sides of the road appear extremely similar making it hard to disambiguate between true-positives and true-



(a) cost matrix C (b) de-correlated cost matrix C' (c) found path

Figure 4.7: Computed similarity matrix from the first iteration in our training process (a). As they contain highly correlated entries, directly finding a matching tour would fail. After the de-correlation of the cost matrix (b) finding a path (c) is significantly more robust.

negatives. As a pre-processing to the tour extraction we remove those correlation effects by subtracting a low-rank matrix approximation

$$C' \leftarrow C - \sum_{k=1}^{r} U_k \Sigma_k V_k^\star \tag{4.4}$$

from singular value decomposition $U\Sigma V^\star$ of C. The result is illustrated in Figure 4.7 (middle) using rank $r = 5$. We found an approximate SVD approach [Vem04] being sufficient. Abusing notation we further denote the de-correlated cost matrix C' as C.

Formulation as a Shortest Path Algorithm

The only missing step for aligning a video pair is to find a plausible path (matching tour) through its respective de-correlated cost matrix. Intuitively, such a path is a collection of consecutive frame-pairs of minimal matching costs over both time dimensions. Matching frame-pairs should lie on a clearly distinct path in the cost matrix. A well-studied algorithm to solve a shortest-path problem is *Dijkstra's Algorithm* [Dij59]. For a given start- and endpoint it computes the globally optimal path with minimum cost. We shortly outline the vanilla grid-version when applying to the cost matrix C.

Dijkstra's algorithm [Dij59]. For detecting paths we assume non-negative costs, *i.e.* $c_{ij} \geq 0$ and only allow for three directions: downwards, rightwards and a diagonal bottom-right step basically preventing reverse playback. Given a start-entry (s, t) and end-point (v, w) with $s \leq v$ and $t \leq w$ the algorithm propagates

costs in $\tilde{C}$ with entries

$$\tilde{c}_{s,t} = 0, (\tilde{c})_{i,j} = \min\{\tilde{c}_{i-1,j}, \tilde{c}_{i-1,j-1}, \tilde{c}_{i,j-1}\} \tag{4.5}$$

$$(c_{\text{direct}})_{i,j} = \arg\min\{\tilde{c}_{i-1,j}, \tilde{c}_{i-1,j-1}, \tilde{c}_{i,j-1}\} \tag{4.6}$$

and infinite costs ∞ for not reachable frames. Following the path backward encoded in (c_{direct}) from (v, w) gives a path P with lowest costs between (s, t) and (v, w). Figure 4.7 c) shows such a path from our augmented version.

Augmented Dijkstra's Algorithm

Unfortunately, we neither know the start- nor end-point in contrast to the vanilla version [Wan+14] nor can we guarantee that there is a path through the entire cost matrix, *e.g.* consider different sub-tours (detours). For most videos of an extensive collection, one does not even know if two videos match at all. We augment Dijkstra's algorithm by processing subsequences individually with the goal to flexibly handle non-matching regions without corrupting the entire path when searching for the global optimum over the entire matrix.

The coarse cost matrix C is split along one time dimension into multiple overlapping column-stripes $C_0, C_1, \ldots, C_n$ (see Figure 4.8) each containing 90 seconds of the video. Now, our approach tries to find a matching tour through the entire cost matrix building on possible tours from each stripe.

Local tours within stripes. Let us consider such a single stripe C_k. Introducing an artificial start node with zero costs to the left enables almost complete freedom regarding the location of each match within one stripe. We are only interested in finding a matching tour from the left $c_{i,0}$ to the right $c_{j,N}$ in the current stripe. Further, this artificial start node allows us to treat each stripe individually.

For the final extraction, we remove path parts from the overlap – taking all information but the overlap – as Dijkstra tends to deviate (see Figure 4.9) from the correct path near the borders of the stripe. For each stripe the vanilla regularized Dijkstra's algorithm is applied as described from S_t to S_{t+1} by only allowing the three mentioned directions. Each possible path $P_t = (S_t, p_1, p_2, \ldots, p_n, S_{t+1})$ has associated matching costs defined as

$$\pi(P_k) = \sum_{p \in P_k} c(p), \tag{4.7}$$

where $c(p)$ is equivalent to an entry c_{ab} in the cost matrix C for the frame-pair $p = (a, b)$.

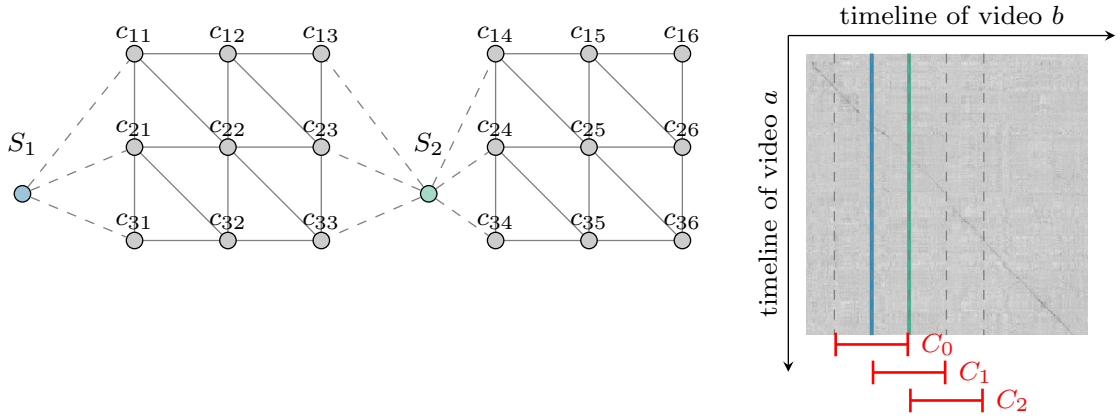


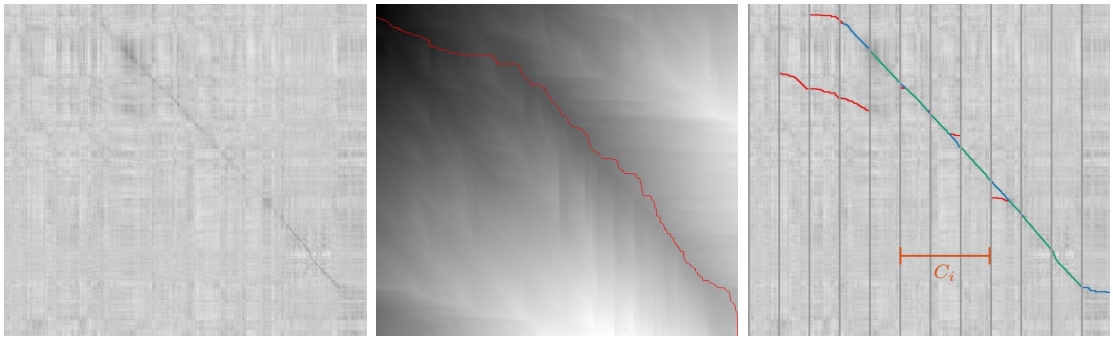
Figure 4.8: Given a cost matrix C, we split up one video by dividing the cost matrix into overlapping stripes $C_0, C_1, \ldots$. Finding the local shortest path in each stripe independently and testing for plausibility in the overlap region, we obtain a reliable matching tour for synchronizing videos.

Heuristic of combining local tours to a global matching tour. As a robust global matching tour does not necessarily span the entire cost matrix and all stripes, *e.g.* for detours, we reject local stripes which cannot be connected to any neighboring stripes with a tolerance of up to two seconds. This tolerance accounts for the fact that we currently only consider coarse information (each 10th frame). Moreover, applying Dijkstra's algorithm to each stripe individually might generate local paths of minimum costs, which are not necessarily a matching tour – there might not exists a matching tour at all. We therefore, reject local stripes with entries $(x, y) \in P_k$ from the local matching tour, if an alternative frame-pair $p' \in \{(x \pm \varepsilon, y), (x, y \pm \varepsilon)\}$ does not have significant higher associated costs than p, *i.e.* we simply use the threshold $c(p') > \xi \cdot c(p)$. Empirical evidence suggests $\xi = \frac{6}{5}$ as a good tradeoff between omitting too many valid frame-pairs and taking frame-pairs erroneously. This threshold might be conservative, but the specific choice only impacts a small fraction of paths during exploration. All remaining frame-pairs from the global matching tour are considered for the generation of the next training dataset version.

4.3.4 Final Alignment of Videos

So far, we only considered each 10th frame during training. In order to prevent temporal miss-alignments due to interpolation artifacts in the final alignment, the full temporal resolution is required.

A coarse-to-fine approach only computes entries of $\bar{C}$ at finer resolution if they are near a matching tour in the coarse cost matrix C, see Figure 4.10. The global



de-correlated cost matrix | propagated costs (Dijkstra's alg.) | union of overlapping stripes

Figure 4.9: For any de-correlated cost matrix C (left) a vanilla Dijkstra-based solution causes severe deviation artifacts and would include frames which are not part of the matching tour following the valley of minimal propagated costs (middle). Our stripe-based heuristic classifies between non-consistent tours-snippets (red, *e.g.* the deviation effect) and reliable tour-snippets (green). The parts illustrated as blue lines will be only taken into account if they do not violate the global consistency constraint.

matching tour is split into chunks of the same size. For each chunk, we compute all frame distances. As the start and end point of the matching tour through a single box is known, we directly apply vanilla Dijkstra's algorithm without further modification.

Modifying the playback speed of only one video would introduce visible jumps and spurts in this video when matching to the reference video. Consider a linear playback of a reference scene with a green traffic light, while the other video has to jump over the frames when waiting on red. To achieve visually pleasing video playback for the human eye, we smooth the matching tour along both time dimensions using Kalman filters [Kal60] with the additional constraint to not revert the timeline of a single video.

4.4 Experiments

We validated our approach on a single workstation with an NVIDIA Titan X GPU. We demonstrate the robustness on aligning a couple of challenging scenes. To evaluate our method despite missing ground truth, we compare our CNN prediction of similarity to [EB11; EB13], the SIFT-based histogram matching from [Wan+14], conducted a user study and evaluate against a manual alignment of videos.

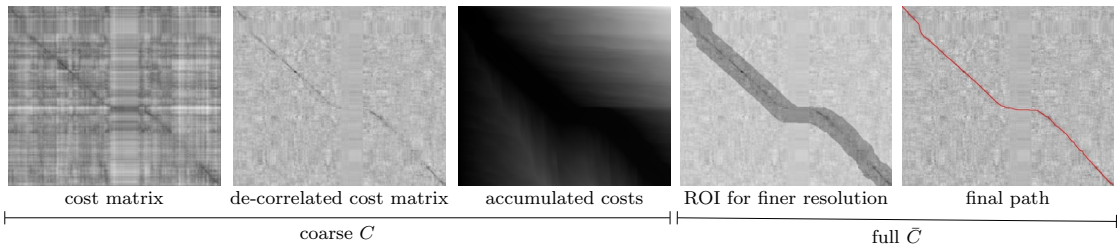


Figure 4.10: Taking each 10th frame into considerations yields a coarse cost matrix C (left). The final path for a frame by frame alignment is just computed on a small region of interest (ROI) in $\bar{C}$ adjacent to the coarse solution.

Table 4.1: Timings for extracting features from unseen videos in fps. Enabling high frame-rates is essential on large-scale datasets.

Approach	[EB11; EB13]	[Wan+14]	[Dou+16]	Ours
Speed	0.11 fps	3.77 fps	15 fps	140 fps

4.4.1 Timings and Memory Consumptions

The presented approach compares favorably to a local-feature-based approach concerning the runtime (Table 4.1) and has small storage requirements (102 MB for the network weights). When considering every 10th frame of a single video with a length of 35 minutes, the encoding takes 45 seconds in total. This allows us to efficiently compute new encodings of the entire dataset of 260 hours content for the training procedure within less than half an hour using 12 GPUs.

The path detection and extraction procedure to produce coarse paths on unseen video-pairs of 35 minutes each takes two seconds given the encodings. This splits into pairwise-distance computation (1071 ms, GPU), de-correlation (370 ms, CPU) and path detection/computation (405 ms, CPU). Computing the final matching tour takes 6 seconds due to multiple runs of the pathfinding procedure on a finer scale. This gives a speed-up factor of at least 300 compared to [Wan+14; EB11; EB13]. So far ours is the first approach enabling large-scale interactive and real-time applications.

4.4.2 Which visual cues are used by the approach?

For aligning videos, the network has to distinguish between relevant and irrelevant regions in the frame, *e.g.* the appearance of traffic and road lanes and the weather depending on the moment of recording. To visualize which input information is used inside the neural network for a particular prediction, we compute saliency

Figure 4.11: Each row contains pairs of coarsely aligned videos (every 10th frame) across different seasons, lightning, weather conditions as well as vegetation. These videos are taken from the validation set and are not used during training. The algorithm can robustly handle windscreen wipers, motion blur and raindrops on the windshield.

maps using guided-ReLU [Spr+15]. Informally, this method computes the gradient information of the network output w.r.t. to the input images holding all weights fixed. Let $f(i) = y$ be a neural network acting as a function approximation with an image i as input and $y \in \mathbb{R}^n$ as an unnormalized classification score with n categories. Then the saliency is $\frac{\partial y'}{\partial i}$ where $y' = \arg\max_j y_j$. The magnitude of this gradient (illustrated in Figure 4.12) indicates pixels where small changes in the input affect the output score most.

We can use these input pixel with high impact on the network prediction, to visualize important features. Comparing the obtained saliency maps of the vanilla ResNet-50 to our trained model (same architecture with different weights) indicates, that our approach learned to ignore irrelevant information like traffic, see Figure 4.12. Instead, it focuses on the shape of the horizon and vegetation of the environment. This is not possible by previous methods [BSH16; ST04; Wan+14; EB11; EB13] using Harris or SIFT features as they also put attention on passing cars and clouds as depicted in the lower row of Figure 4.12.

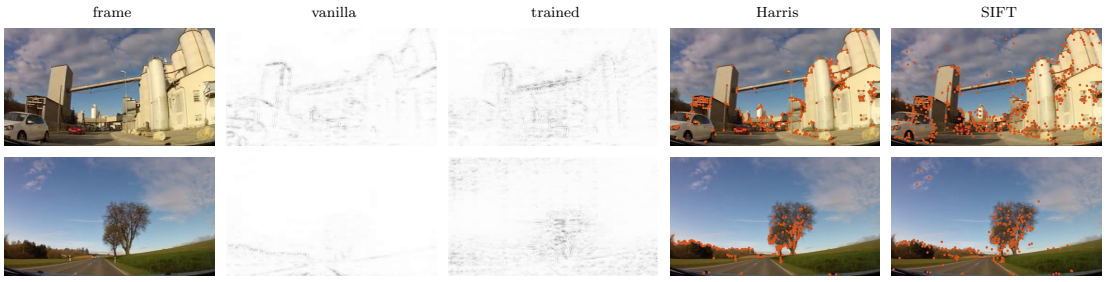


Figure 4.12: The network (trained) learned to ignore common content such as traffic or road lanes and instead focuses on striking environment information compared to vanilla ResNet. Previous methods [BSH16; ST04; Wan+14; EB11; EB13] rely on local features (Harris Corner Feature Detector (Harris), SIFT), capturing irrelevant information like the white car or clouds.

4.4.3 Robustness and Accuracy

Unlike methods based solely on aggregating SIFT vectors, our method is able to synchronize even videos recorded five months apart as depicted in Figure 4.14. Remarkably, as the videos for the dataset are collected over multiple months, the network has learned to interpret scenes globally. Even sequences where human interventions like tree-felling cause a rather different look of the same scene, the respective videos are immaculately synchronized by our approach as we enforce temporal consistency.

To evaluate the accuracy quantitatively, we manually assessed 500 tours predicted by our approach from the cost matrices (see Figure 4.5, left). Note, how the accuracy increases over the iterations. Hence, the harvested additional training data of higher complexity results in a higher recall of found matching tours.

In addition, an expert annotated the videos of Figure 4.11 for two experiments thoroughly by manually adjusting the best visual matching frame. We regard this as expert annotation, since the locations shown are known to the annotator and the temporal context can be studied by playing the video forwards or backwards. In a user study, we displayed a single reference frame and the manually aligned frames besides several other neighboring frames. The evaluation of 450 submitted results from 14 participants is illustrated in Figure 4.15, which reveals discordance between different participants on the same frame. Approximately, 53% agree with a tolerance of four frames. This clearly reveals the difficulty of this task, which is presumably caused by a change of perspective of the camera, lack of temporal information in rural scenes or unfamiliarity with the places shown. This might also contain challenging examples as shown in Figure 4.13. Consequently, we directly compared the frame-distance between our annotations and the extracted path of

Figure 4.13: The left frame was captured in November 2012 and right frame in March 2013 after tree felling works. These kinds of differences cause high matching costs in the found tour in our approach, but the situation is resolved by our matching tour heuristic utilising temporal consistency.

our approach. This compares favorably to our estimated human performance as 62% of the predicted frame-pairs have a frame offset of at most four frames, see Figure 4.15.

4.4.4 Multi-Video Alignment

Having all coarse encodings Φ_i at hand, we can query multiple videos showing the same scene independent from appearance and simultaneously align them accordingly. Frames from the synchronized video snippets are illustrated in Figure 4.16, which shows the alignment of 7 videos across different seasons. The reference video is represented by the left-most frames. Although these videos feature all deviations from the reference video in terms of weather and seasonal effects, cloudiness, illumination and windshield wipers the alignment stays robust — even over longer temporal range. For efficiency reasons, the nearest neighbor lookup is restricted to take only each 10th frame encoding into account.

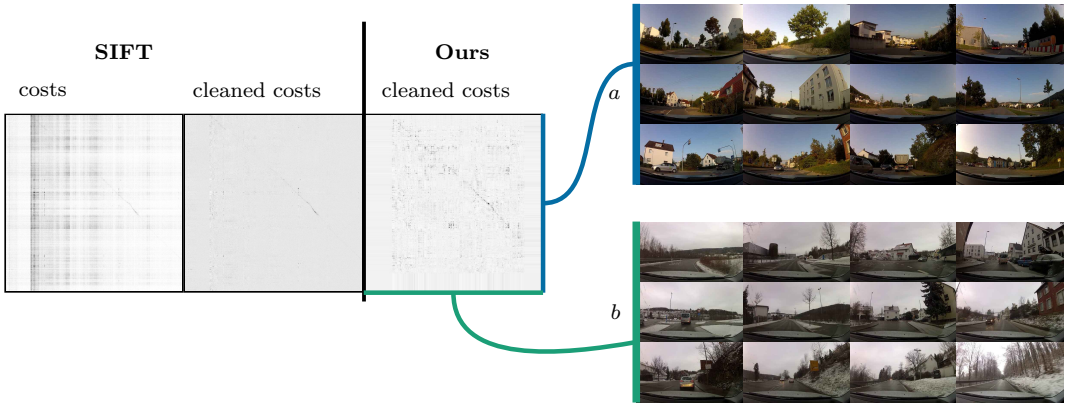


Figure 4.14: Although, there should be a matching tour between the videos a and b (taken in September 2012 and February 2013 respectively) through the entire video snippet, it is not possible to align both videos using SIFT vectors [Wan+14]. The cost matrix in our approach contains reasonable information for most frames.

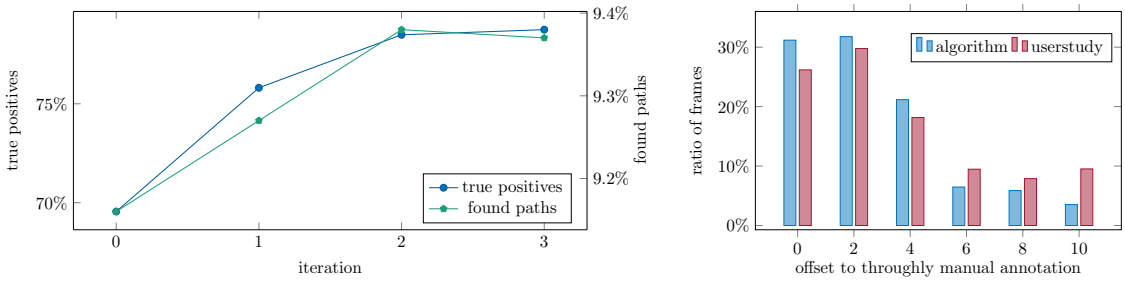


Figure 4.15: Comparison to human annotated alignment. After each iteration we report (left) the number of found matching tours between videos and true positives (expert annotation). The mis-alignment from participants in a user-study and the automatic results of our approach is evaluated against ground-truth from the expert annotation (right).

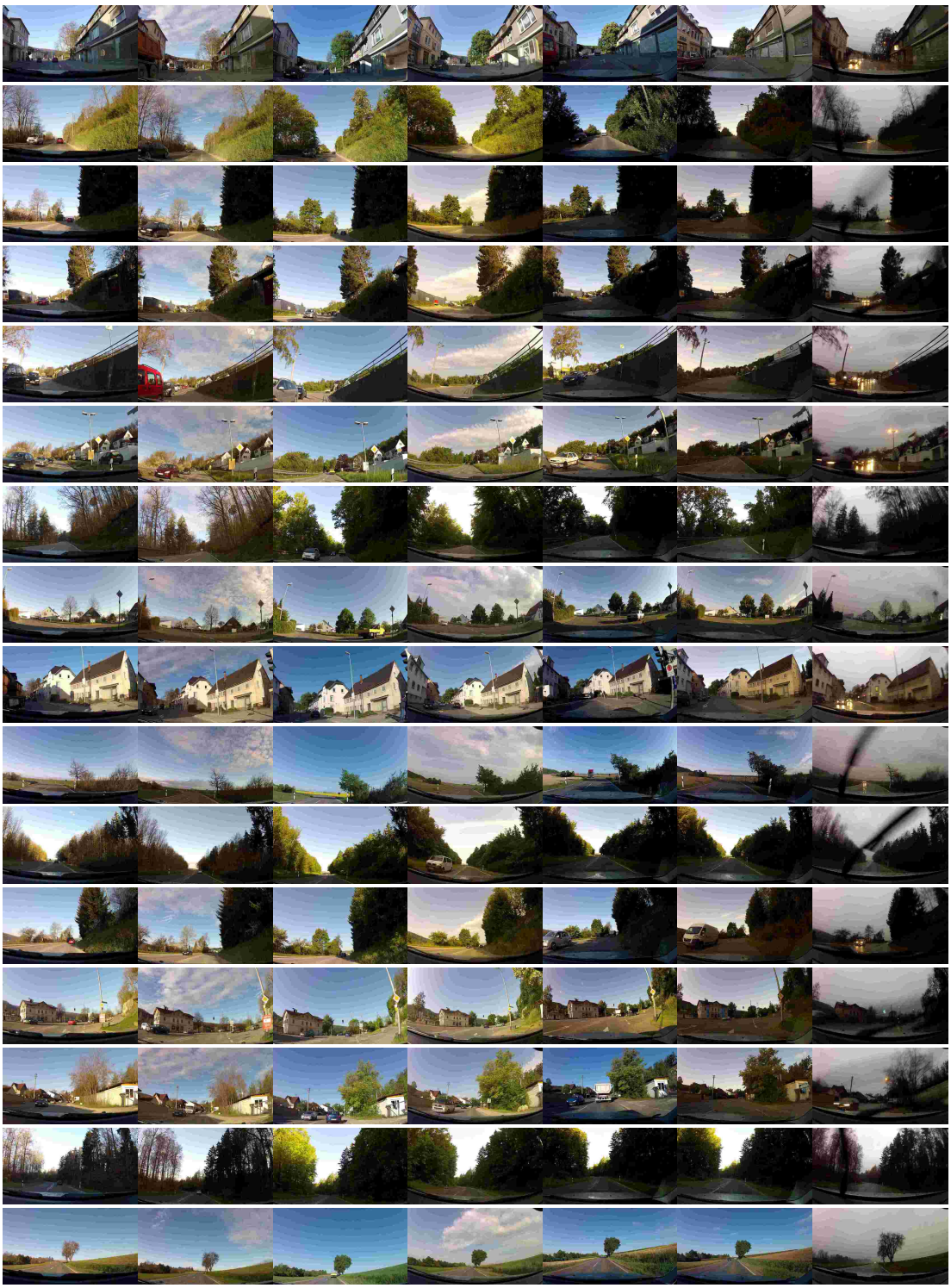


Figure 4.16: Multi-Video *temporal* alignment: After querying similar and time consistent video snippets using the learned encodings, the approach is able to robustly synchronize all found snippets to a reference video snippet.

4.5 Conclusion

As the amount of available data increases, algorithms based on supervised training will not catch up to integrate all the data when the manual labeling process remains the bottleneck. During this chapter, a training method for deep neural networks was developed, which improved the performance of neural networks without human interaction in the loop and without requiring any label.

On the basis of a heuristic to iteratively generate labels for certain training data our approach effectively demonstrates that a large amount of multi-frame data represents an opportunity if we can take advantage natural properties that this data features (temporal consistency) as opposed to single-frame observations. Such properties can act as a lever to control the entire learning process if they are used during training. Further, taking advantage of heuristic properties like transitivity training data can be gathered which does not arise originally from the heuristic itself (frame-pairs from videos which are captured months apart under different weather conditions and vegetation).

The presented approache learned an organization of all data by structuring the underlying dataset in a latent space by distinguishing relevant and irrelevant features. This concludes that in addition to the optimization procedure of how to adjust the learnable parameters in a deep neural network the sampling process of the training data is important as well. Instead of looping over a random permutation of the training set, a more sophisticated sampling of the training data has meaninful impact.

4.6 Possible Future Work

In the previous sections, we introduced a method to automatically establish a link between images that feature the same scenery but differ in appearance. Being able to query multiple such frame-pairs opens new possibilities to learn photorealistic image-to-image translations [Iso+17], which is currently constrained by the availability of training data in high quality as it mostly relies on synthetic data gathered from modern games [Ric+16]. Image-to-image translation methods based on Generative Adversarial Networks (GANs), which learn a mapping $f\colon \mathcal{A} \to \mathcal{B}$ between two sets of images $\mathcal{A}, \mathcal{B}$ are usually limited by restrictions such as knowing the exact matches $\{(a, b), a \in \mathcal{A}, b \in \mathcal{B}, f(a) = b\}$ in the training set or they are indeed limited to only two (disjoint) sets [Zhu+17]. Extensions [Hua+18] are able to produce a variety of generated images from multiple sets, but are limited to cases where these sets are rigorously separable by well-defined and known boundaries in simple cases[4], such as “cats” to “dogs” or “summer” to “winter” and vice versa [Hua+18].

[4] An implicit assumption made in these attempts is that the latent coding has a categorical distribution.

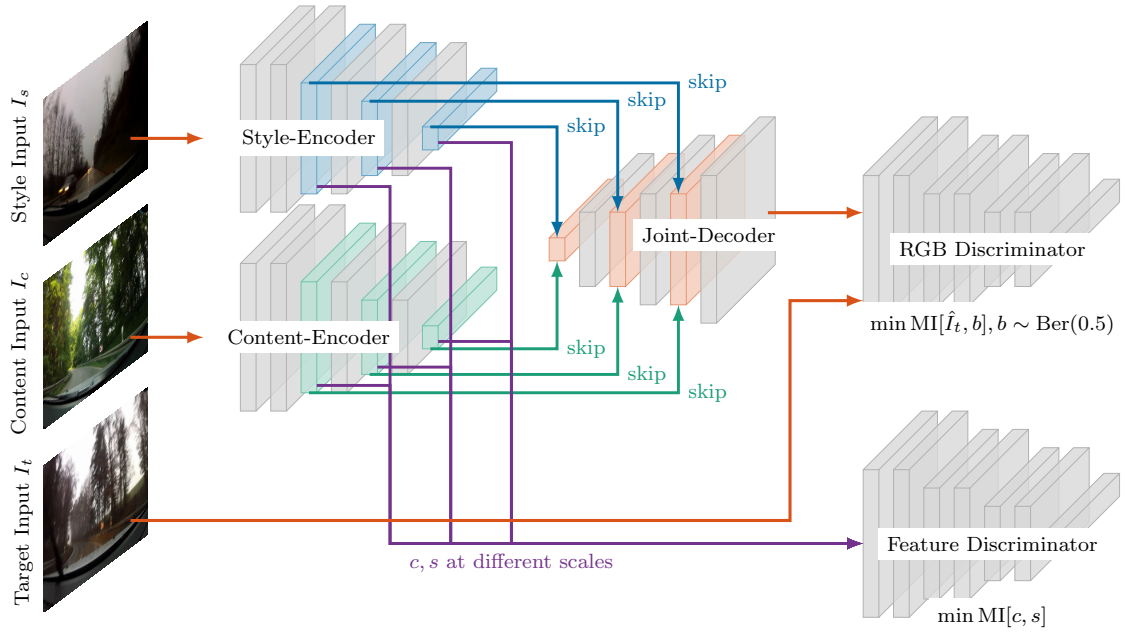


Figure 4.17: Given triplets of content, style and target images the joint decoder is trained to synthesize photorealistic images (RGB Discriminator) wrt. Mutual Information (MI) between $\hat{I}_t$ and some binary coin-flip $b \sim$ Ber(0.5). The additional features discriminator enforces small Mutual Information MI$[c, s]$ even when extracted from the same image.

In our case, the matching tour contains orthogonal information to what has been used by known methods in literature [Iso+17; Zhu+17; Hua+18]. Each entry of the matching tour can provide exact matches (x_i, x_j) from subsets $\mathcal{X}_i, \mathcal{X}_j \in 2^{\mathcal{F}}$ of all frames, which however are not necessary disjoint in terms of appearance, *e.g.* $\mathcal{X}_i$ might represent all frames captured in winter and $\mathcal{X}_j$ all videos showing dusk. In our case, there is no clear distinction possible and the samples are distributed continously in the latent space instead of merely belonging to class $\mathcal{A}$ or $\mathcal{B}$. Given such image analogies, training a neural network for video editing is possible such that a video appearance can be altered during playback.

Given a triplet of a content image I_c, style image I_s and target image I_t a possible attempt is to trained two encoder networks E_c, E_s to encode content $c = E_c(I_c)$ as well as style information $s = E_s(I_s)$, see Figure 4.17. The joint decoder network $D(c, s)$ is trained to fit a target image I_t in adversarial fashion by minimizing the MI (MI$[x, b]$ between a coin-flip outcome b and x). Hereby x is either $\hat{T}_t$ or T_t depending on the actual value of b. The entire pipeline can be considered as a latent-variable model, which disentangles the content representation and style information — especially when an additional Discriminator forces small MI$[c, s]$ between extracted features c and features s. This can be done by training a discriminator to decide

Figure 4.18: Transferred appearances from style-inputs (single images) applied to content inputs (frames of one video).

whether s, c are extracted from the same image or not.

Any such trained model can be used to augment training data for *e.g.* semantic segmentation task. Preliminary results of the synthesizing process are depicted in Figure 4.18. Further, any interpolation in style $s = \lambda s_1 + (1 - \lambda)s_2$ between two styles s_1, s_2 is likely to cause such a smooth translation in RGB space from the decoder output. It has been already demonstrated [Lar+16], that these latent spaces are highly structured and therefore simple latent vector space calculus leads to complex operations on the generated images. It remains open, how to train a network to interpolate a specific effect, *e.g.* weather change or seasons change. The InfoGAN [Che+16] approach has demonstrated such abilities in shaping the learned latent space representation without any annotation – but as of yet only for MNIST digits.

Rule #7:
Deep learning is highly forgiving.
Although GANs are a disaster in theory,
they deliver amazing results in practice!

Chapter 5

Separating Reflection and Transmission Images in the Wild

Reflections caused by common semi-reflectors, such as glass windows, can severely impact the performance of computer vision algorithms. Previous works can remove reflections from synthetically generated data or in controlled scenarios. However, they are based on strong assumptions and do not generalize well to real-world images. Contrary to a common misconception, real-world images are challenging even when physical effects like polarization of light are used. We present a deep learning approach to separate the reflected and the transmitted components of the recorded irradiance, which explicitly uses the polarization properties of light in a multi-frame setting.

The material of this chapter is based on the following publication:

[Wie+18] Patrick Wieschollek, Orazio Gallo, Jinwei Gu, and Jan Kautz. „Separating Reflection and Transmission Images in the Wild“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018

In the previous chapter, we witnessed the ability of neural networks to deal with different appearances in a single-value discriminative approach (deciding whether two frames show the same scene or not). However, depth estimation from observations generally necessitates a per-pixel estimation and therefore relies on a model capable of generating such information. In this chapter, we present a learning-based method generating a per-pixel estimate as a pre-processing step for subsequent pipelines. The motivation for this pre-processing is a result of the assumption made by many computer vision algorithms: The value of each observed pixel is a function of the radiance of a single area in the scene. Semi-reflectors, such as typical windows or glass doors, regularly break this assumption by creating a superposition of the radiance of two different objects: the one behind the surface (transmission) and the one that is reflected (reflection). Now, seeking for corresponding pixels from multiple views requires to deal with such ambiguities, see Figures 5.1 (a,b).

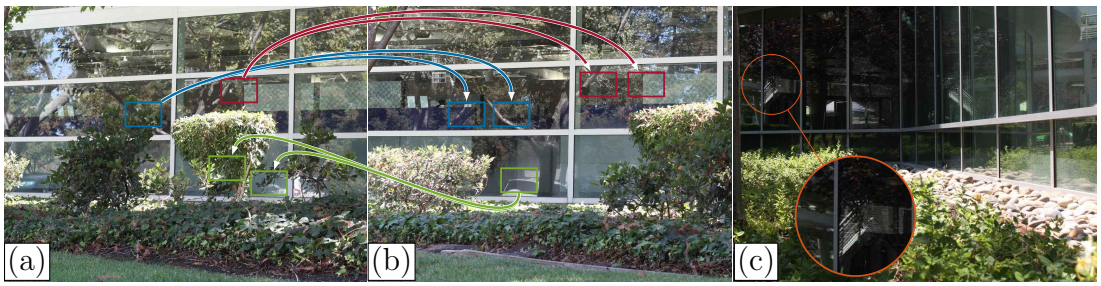


Figure 5.1: Correspondence issues when matching patches between the observations (a) and (b). Any 3D reconstruction algorithm relying on matching corresponding pixels resp. patches will need to disambiguate between two possible candidate patches matches respecting either information of reflection or transmission. Further, note how the reflection in the window facade manifests information (c) about the scene, which is not directly observable.

Here, each highlighted patch can be mapped to two candidate patches in the second view depending on whether the choice is made on the reflection or transmission information.

It is virtually impossible to avoid semi-reflectors in human-made environments, as can be seen in Figure 5.2(a), which shows a typical downtown area. This effect can impact tasks such as recognition or 3D reconstruction by introducing noise or even causing them to fail. As a dense 3D reconstruction relies on matching pixel from different views, any multi-view stereo or SLAM algorithm would be hard-pressed to produce accurate reconstructions on this type of images. This is where a trained neural network for pre-processing comes into play.

Several methods exist that attempt to separate the reflection and transmission layers. At a semi-reflective surface, the observed image, I_o, can be modeled as a linear combination of the reflection and the transmission images:

$$I_o = \alpha_r I_r + \alpha_t I_t. \tag{5.1}$$

Equation 5.1 shows that the problem is ill-posed: we need to estimate multiple unknowns from a single observation. A solution, therefore, requires additional priors or data. Indeed, previous works heavily rely on assumptions about the appearance of the reflection (*e.g.*, it is blurry when captured with large aperture and focusing to infinity), about the shape and orientation of the surface (*e.g.*, it is perfectly flat and exactly perpendicular to the principal axis of the camera), and others. Images taken in the wild, however, regularly break even the most basic of these assumptions, see Figure 5.2 (b), causing the results of state-of-the-art methods [SSK00; KTS; Fan+17a] to deteriorate even on seemingly simple cases, as

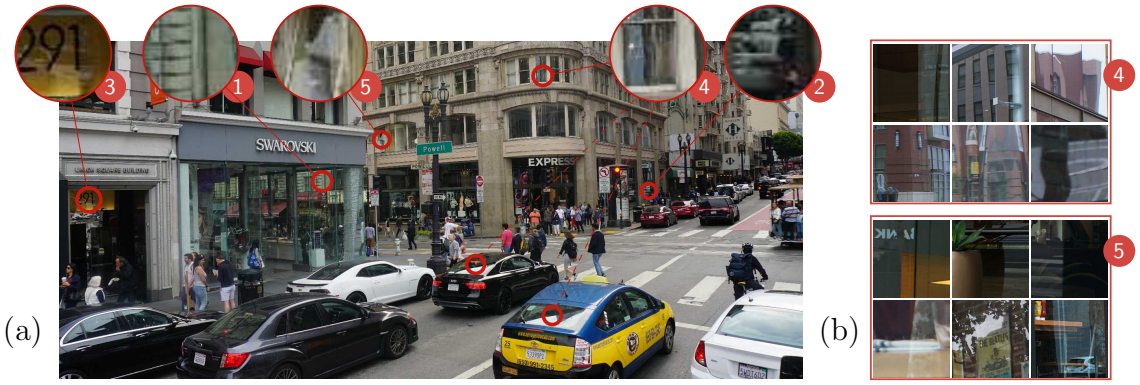


Figure 5.2: Depending on the ratio between transmitted and reflected radiance, a semi-reflector may produce no reflections ❶, pure reflections ❷, or a mix of the two, which can vary smoothly ❸, or abruptly ❺. The local curvature of the surface can also affect the appearance of the reflection ❹. The last two, ❹ and ❺, are all but uncommon, as shown in (b).

shown in Figure 5.3, which depicts a fairly typical real-world scene.

However, a successful separation between both components facilitates reconstructions using scene information which is never directly observed. The footbridge and stairs in Figure 5.1 (c) are not directly visible from the current camera orientation, but the manifestation at the glass front might still convey enough information for proper reconstruction. Hence, a successful separation between reflection and transmission might guide, complete or in the first place concede 3D reconstructions.

One particularly powerful tool to address the ill-posedness of the problem is polarization: when unpolarized light interacts with an interface separating media with different refractive indexes, its transmitted and reflected components take on different polarization states. Hence, images captured through a polarizer oriented at different angles offer additional observations.

An approach requiring images captured under different polarization angles obviously requires corresponding hardware. Cameras that can simultaneously capture multiple polarization images exist[1] and have the potential to become more popular, at least for professional applications. Indeed, methods based on polarization are not intended to benefit a casual photographer. Rather, they are geared towards popular computer vision applications such as content capture for virtual reality and gaming, or aforementioned 3D reconstruction. For such applications, which are hindered by reflections, specialized hardware and specific capturing procedures are generally considered an acceptable cost.

Our analysis of the state-of-the-art methods indicates that the quality of the

[1]See, for instance, `https://www.ricoh.com/technology/tech/051_polarization.html` or `http://www.fluxdata.com/imaging-polarimeters`.

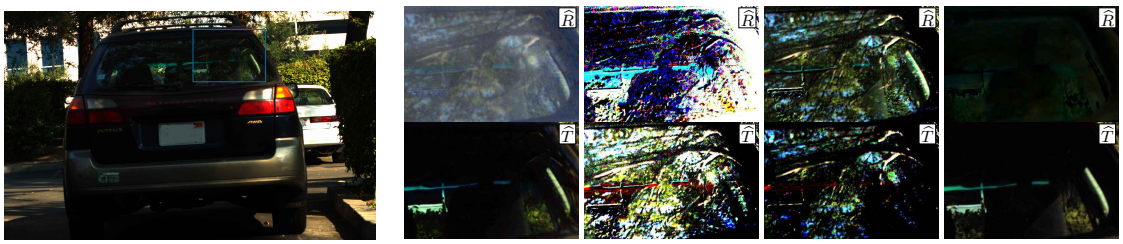


Figure 5.3: Glass surfaces are virtually unavoidable in real-world pictures. Our approach to separate the reflection and transmission layers, works even for general, curved surfaces, which break the assumptions of state-of-the-art methods. In this example, only our method can correctly estimate both reflection (the tree branches) and transmission (the car's interior).

obtained results degrades significantly when moving from synthetic to real data, even when using polarization. This is due to the simplifying assumptions that are commonly made, but also to an inherent issue that is all too often neglected: A polarizer's ability to attenuate reflections dramatically depends on the viewing angle [Col05]. The attenuation is maximal at an angle called the Brewster[2] angle, θ_B. However, even when part of a semi-reflector is imaged at θ_B, the Angle Of Incidence (AOI) in other areas is sufficiently different from θ_B to essentially void the effect of the polarizer, as clearly shown later in Figure 5.5. In other words: Because of limited signal-to-noise ratio, for specific regions in the scene, the additional observations may not be independent.

Since even polarization information is not always sufficient, we propose to complement it with the ability to learn prior information directly from images. We present a deep-learning method capable of separating the reflection and transmission components of images captured in the wild. Our approach does not make assumptions about the curvature of semi-reflecting surfaces (including the local curvature variations caused by the manufacturing process), and can deal with considerable variations of dynamic range between the reflected and the transmitted scene, and with non-rigid scene motion, both of which are severe issues for polarization-based methods. The success of our method stems from our two main contributions. First, rather than requiring a network to learn the reflected and transmitted images directly from the observations, we leverage the properties of light polarization and use a residual representation, and a layer that projects the input images on the canonical polarization angles (Section 5.2.1 and 5.2.2). Second, ground truth data is not available and needs to be synthesized. Therefore, we design an image-based data

[2]For window panes Snell's law dictates $\theta_B = \arctan\left(\frac{n_2}{n_1}\right) \approx 56.31°$ with refactive indicies $n_1 = 1, n_2 = 1.5$ for air respectively glass.

generator that faithfully reproduces the image formation model (Section 5.2.3). We show that the different parts of the proposed pipeline are indeed necessary to model real-world behaviors. Notably, our method can successfully separate the reflection and transmission layers even in challenging cases, on which previous works does not meet the desired result. To further validate our findings, we capture the Urban Reflections Dataset, a polarization-based dataset of reflections in urban environments that can be used to test reflection removal algorithms on realistic images.

5.1 Related Work

There is a rich literature of methods dealing with semi-reflective surfaces, which can be organized in three main categories based on the underlying assumptions.

Single-image methods can leverage gradient information to solve the problem. Levin and Weiss, for instance, require manual input to separate gradients of the reflection and the transmission [LW07]. Methods that are fully automated can distinguish the gradients of the reflected and transmitted images by leveraging the defocus blur [LB14]: reflections can be blurry because the subject behind the semi-reflector is much closer than the reflected image [Fan+17a], or because the camera is focused at infinity and the reflected objects are close to the surface [ADAS17]. Moreover, for the case of double-pane or thick windows, the reflection can appear "doubled" [DS08], and this can be used to separate it from the transmitted image [Shi+15]. While these methods show impressive results, their assumptions are stringent and do not generalize well to real-world cases, causing them to fail in common real-world scenarios.

Multiple images captured from different viewpoints can also be used to remove reflections. Several methods propose different ways to estimate the relative motion of the reflected and transmitted image, which can be used to separate them [LB13; Xue+15; SAA00; GCM14; HS17]. It is important to note that these methods assume static scenes—the motion is the apparent motion of the reflected layer relative to the transmitted layer, not scene motion. Other than that, these methods make assumptions that are less stringent than those made by single-image methods. Nonetheless, these algorithms work well when reflected and transmitted scenes are shallow in terms of depth, so that their velocity can be assumed uniform. For the case of spatially and temporally varying mixes, Kaftory and Zeevi [KZ13] propose to use sparse component analysis instead.

Multiple images captured under different polarization angles offer a third venue to tackle this problem. Unpolarized light takes different polarization states depending on whether it is transmitted or reflected by a semi-reflector. One can separate between reflection and transmission assuming that images taken at different polarization angles offer independent measurement of the same scene by using independent component analysis (ICA) [FA99; BYO+01; Bro+05]. Another

way to separate reflection from transmission exploits semi-reflective surface which generates double reflections in combination with polarization information [DS08]. Under ideal conditions, and leveraging polarization information, a solution can also be found in closed form [SSK00; KTS]. In our experiments, we found that most of the pictures captured in unconstrained settings break even the well-founded assumptions used by these papers, as shown in Figure 5.2.

To allow our approach to work on images captured in the wild, we avoid making assumptions about reflections altogether. We rather leverage polarization in a learning-based approach with constraints offered by the properties of polarized light, and propose a novel approach to synthesizing realistic training data.

5.2 Method

We address the ill-posed problem of layer decomposition, by leveraging the ability of a semi-reflector to polarize the reflected and transmitted layers differently. Capturing multiple polarization images of the same scene, offers partially independent observations of the two layers. To learn to use these observations to predict the latent layers directly, we use an encoder-decoder neural network. Since the ground truth for this problem is virtually impossible to capture, our only option is to synthesize it. As for any data-driven approach, the realism of the training data is paramount to the quality of the results, which makes it critical to design a data-generation pipeline that captures the many nonidealities affecting real data.

In this section, after reviewing the image formation model, we give an overview of our approach and we discuss the limitations of the assumptions that are commonly made. Moreover, we describe how we address them in our data generation pipeline. Finally, we describe the details of our implementation.

5.2.1 Polarization, Reflections and Transmissions

Consider two points, P_R and P_T such that the reflection $P_R^{'}$ of P_R, lies on the line of sight of P_T, and assume that both emit unpolarized light, see Figure 5.4. After being reflected or transmitted, unpolarized light becomes polarized by an amount that depends on θ, the AOI.

At point P_S, the intersection of the line of sight and the surface, the total radiance $L(P_S, \theta)$ is a combination of the reflected radiance $L_R(P_S, \theta)$, and the transmitted radiance $L_T(P_S, \theta)$. If we place a linear polarizer with polarization angle ϕ in front of the camera, and we average out the solid angle and exposure time to get from the radiance at P_S the corresponding intensity at pixel x on the camera sensor, we can write for each pixel x:

$$I_\phi(x) = \alpha(\theta, \phi_\perp, \phi) \cdot \frac{I_R(x)}{2} + (1 - \alpha(\theta, \phi_\perp, \phi)) \cdot \frac{I_T(x)}{2}, \tag{5.2}$$

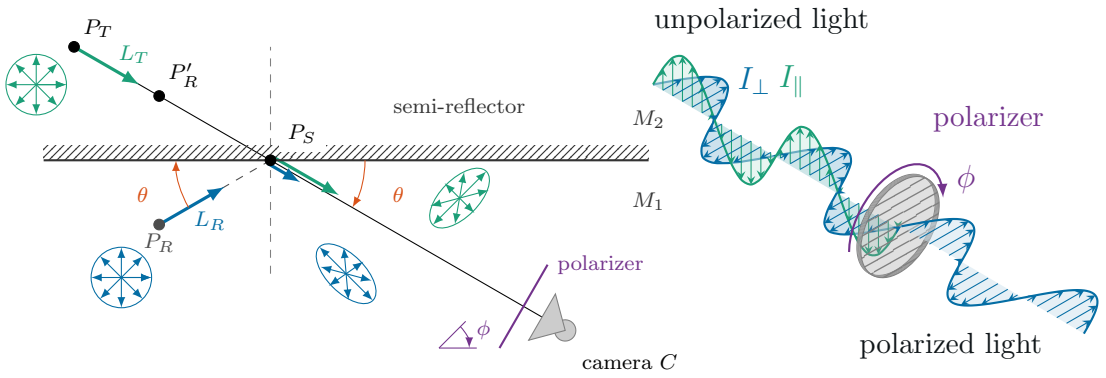


Figure 5.4: Unpolarized light from P_T and P_R becomes polarized at P_S by a degree that is a function of the AOI, θ. Camera C observes the superposition of the radiance of objects P_T and P'_R, the reflection of P_R. Depending on the amount of polarization and on the angle of the polarizer, ϕ, the observation changes.

where $\alpha(\cdot) \in [0, 1]$ denotes the mixing coefficient, $\theta(x) \in [0, \pi/2]$ denotes the AOI, $\phi_\perp(x) \in [-\pi/4, \pi/4]$ describes the $p-$polarization direction [SSK00], and $I_R(x)$ resp. $I_T(x)$ the reflected resp. transmitted images at the semi-reflector. All terms on the right side of Equation (5.2) unknown.

At the Brewster angle, θ_B, the reflected light is completely polarized along $\phi_\perp$, *i.e.* in the direction perpendicular to the incidence plane[3], and the transmitted light along $\phi_\parallel$, the direction parallel to the plane of incidence. The angles $\phi_\perp$ and $\phi_\parallel$ are called the canonical polarization angles. In the unique condition in which $\theta(x) = \theta_B$, two images captured with the polarizer at the canonical polarization angles offer independent observations that are sufficient to disambiguate between I_R and I_T. Unless the camera or the semi-reflector are at infinity, however, $\theta(x) = \theta_B$ only holds for few points in the scene, if any. Figure 5.5 shows a typical case where part of the reflection is imaged around θ_B (the reflection of the plant), while the reflection of a nearby object, the book on the right, is virtually unaffected by the different polarization angles.

To complicate things, for curved surfaces $\theta(x)$ depends on x in non-linear way. Making any assumptions about the AOI can lead to severe artifacts in the separation of I_R and I_T. Finally, even for arbitrarily many acquisitions at different polarization angles, ϕ_j, the problem remains ill-posed as each observation I_{ϕ_j} adds new pixel-wise unknowns $\alpha(\theta, \phi_\perp, \phi_j)$.

[3]The incidence plane is defined by the direction, in which the light is traveling and the semi-reflector's normal.

Figure 5.5: A polarizer attenuates reflections when they are viewed at the Brewster angle. For the scene shown on the left, we manually selected the two polarization directions that maximize and minimize reflections respectively. Indeed, the reflection of the plant is almost completely removed. However, only a few degrees away from the Brewster angle, the polarizer has little to no effect, as is the case for the reflection of the book on the right.

5.2.2 Recovering R and T

When viewed through a polarizer oriented along direction ϕ, I_R and I_T, which are the reflected and transmitted images at the semi-reflector, produce image I_ϕ at the sensor. Due to differences in dynamic range, as well as noise, in some regions the reflection may dominate I_ϕ, or vice versa, see Section 5.2.3. Without hallucinating content, one can only aim at separating the original sources R and T, which we define to be the observable reflected and transmitted components. For instance, T may be zero in regions where R dominates, even though I_T may be greater than zero in those regions. To differentiate them from the ground truth, we refer to our estimates as $\widehat{R}$ and $\widehat{T}$.

To recover $\widehat{R}$ and $\widehat{T}$, we use an encoder-decoder architecture, which has been shown to be particularly effective for many tasks, such as image-to-image translation [Iso+17] or denoising [MSY16a]. Learning to estimate $\widehat{R}$ and $\widehat{T}$ directly from images taken at arbitrary polarization angles does not produce satisfactory results. One main reason is that parts of the image may be pure reflections, thus yielding no information about the transmission, and vice versa.

To address this issue, we turn to the polarization properties of reflected and transmitted images. Recall that R and T are maximally attenuated, though generally not completely removed, at $\phi_\parallel$ and $\phi_\perp$ respectively. The canonical polarization angles depend on the geometry of the scene, and are thus hard to capture directly. However, we note that an image $I_\phi(x)$ can be expressed as [KTS]:

$$I_\phi(x) = I_\perp(x)\cos^2(\phi - \phi_\perp(x)) + I_\parallel(x)\sin^2(\phi - \phi_\perp(x)). \tag{5.3}$$

Since Equation (5.3) has three unknowns, $I_\perp$, $\phi_\perp$, and $I_\parallel$, we can use three different

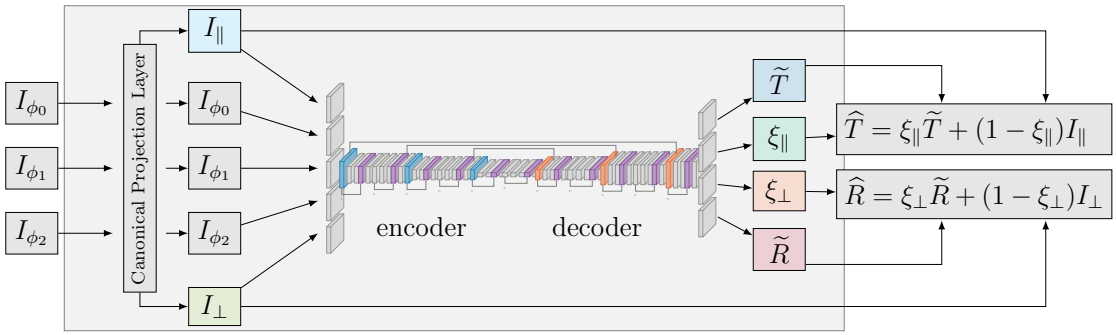


Figure 5.6: Our encoder-decoder network architecture with ResNet blocks includes a Canonical Projection Layer, which projects the input images onto the canonical polarization directions, and uses a residual parametrization for $\widehat{T}$ and $\widehat{R}$.

observations of the same scene, $\{I_{\phi_i}(x)\}_{i=\{0,1,2\}}$, to obtain a linear system that allows to compute $I_{\perp}(x)$ and $I_{\parallel}(x)$. To further simplify the math we capture images such that $\phi_i = \phi_0 + i \cdot \pi/4$.

For efficiency, we implement the projection onto the canonical views as a network layer in TensorFlow, see Figure 5.6. The canonical views and the actual observations are then stacked in a 15-channel tensor and used as input to our network. Then, instead of training the network to learn to predict $\widehat{R}$ and $\widehat{T}$, we train it to learn the residual reflection and transmission layers. More specifically, we train the network to learn an 8-channel output, which comprises the residual images $\widetilde{T}(x)$, $\widetilde{R}(x)$, and the two single-channel weights $\xi_{\parallel}(x)$ and $\xi_{\perp}(x)$. Dropping the dependency on pixel x for the sake of clarity, we can then compute:

$$\widehat{R} = \xi_{\perp}\widetilde{R} + (1 - \xi_{\perp})I_{\perp} \qquad \text{and} \qquad \widehat{T} = \xi_{\parallel}\widetilde{T} + (1 - \xi_{\parallel})I_{\parallel}. \tag{5.4}$$

While $\xi_{\perp}$ and $\xi_{\parallel}$ introduce two additional unknowns per pixel, they significantly simplify the prediction task in regions where the canonical projections are already good predictors of $\widehat{R}$ and $\widehat{T}$. We use an encoder-decoder with skip connections [RFB15] that consists of three down-sampling stages, each with two ResNet blocks [He+16b]. The corresponding decoder mirrors the encoding layers using a transposed convolution with two ResNet blocks. We use an ℓ_2 loss on $\widehat{R}$ and $\widehat{T}$. We also tested ℓ_1 and a combination of ℓ_1 and ℓ_2, which did not yield significant improvements.

The use of the canonical projection layer, as well as the parametrization of residual images is key to the success of our method. Figure 5.7 depicts a comparison our network with the output of the exact same architecture trained to predict $\widehat{R}$ and $\widehat{T}$ directly from the three polarization images $I_{\phi_i}(x)$.

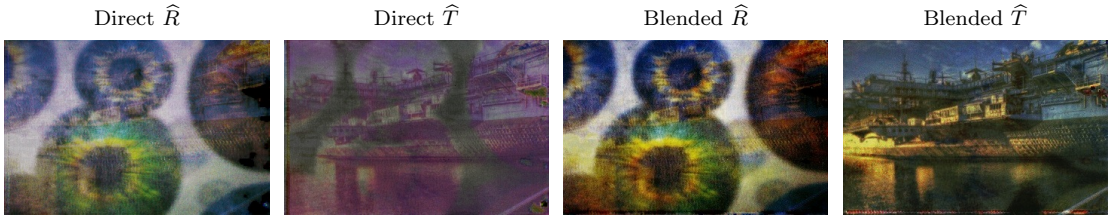


Figure 5.7: Comparison of outputs from the same network architecture either trained to directly estimate $\widehat{R}$ and $\widehat{T}$ or to predict the residuals as propose in this chapter.

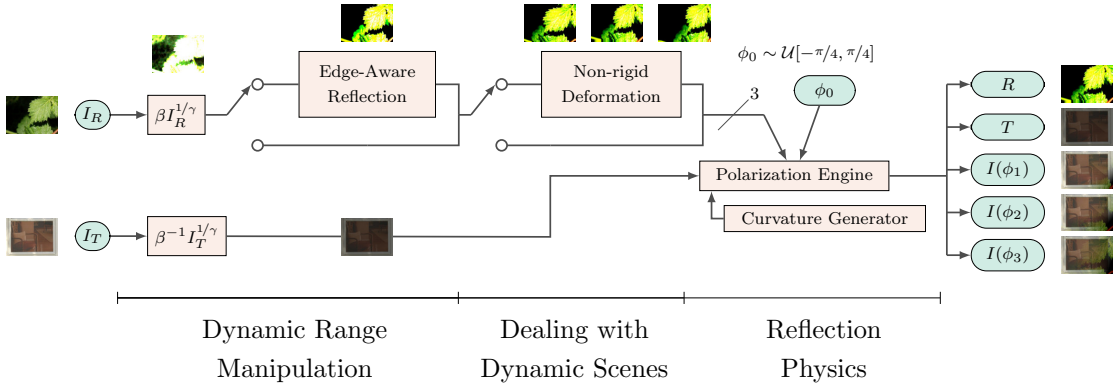


Figure 5.8: The image-based data generation procedure. We apply several steps to images I_R and I_T simulating reflections in most real-world scenarios (Section 5.2.3). Starting by sampling I_R, I_T from an image database we apply several steps to simulate reflections in most real-world scenarios.

5.2.3 Image-based Data Generation

The ground truth data to estimate $\widehat{R}$ and $\widehat{T}$ is not available and virtually inaccessible to capture perfectly. Recently, Wan *et al.* released a dataset for single-image reflection removal [Wan+17], but it does not offer polarization information. One could synthesize it by means of ray tracing, but most of the available rendering engines do not model polarization. In principle, Equation 5.2 could be used directly to generate the data we need from any two images. The term α in the equation, however, hides several subtleties and nonidealities. For instance, previous polarization-based works use it to synthesize data by assuming uniform AOI, perfectly flat surfaces, comparable power for the reflected and transmitted irradiance, or others. This generally translates to poor results on images captured in the wild: Figures 5.2 and 5.3 show common scenes that violate all of these assumptions. Those difficulties are even out-of-scope for the patch-wise extension by Kong *et al.* [KTS].

We propose a more accurate synthetic data generation pipeline, see Figure 5.8. Our pipeline starts from two randomly picked images from the PLACE2 dataset [Zho+17], I_R and I_T, which we treat as the image of reflected and transmitted scene at the surface. From those, we model the behaviors observed in real-world data, which we describe as we "follow" the path of the photons from the scene to the camera.

Dynamic Range Manipulation at the Surface

To simulate realistic reflections, the Dynamic Range Manipulation (DR) of the transmitted and reflected images at the surface must be significantly different. This is because real-world scenes are generally high-dynamic-range (HDR). Additionally, the light intensity at the surface drops with the distance from the emitting object, further expanding the combined DR. However, our inputs are low-dynamic-range images because a large dataset of HDR images is not available. We propose to artificially manipulate the DR of the inputs to match the appearance of the reflections we observe in real-world scenes.

Going back to Figure 5.4, we note that for regions where $L_T \approx L_R$, a picture taken without a polarizer will capture a smoothly varying superposition of the images of P_R and P_T (Figure 5.2 ❸). For areas of the surface where $L_R \gg L_T$, however, the total radiance is $L \approx L_R$, and the semi-reflector essentially acts as a mirror (Figure 5.2 ❷). The opposite situation is also common (Figure 5.2 ❶). To allow for these distinct behaviors, we manipulate the dynamic range of the input images with a random factor $\beta \sim \mathcal{U}[1, K]$:

$$\tilde{I}_R = \beta I_R^{1/\gamma} \qquad \text{and} \qquad \tilde{I}_T = \frac{1}{\beta} I_T^{1/\gamma}, \tag{5.5}$$

where $1/\gamma$ linearizes the gamma-compressed inputs[4]. We impose that $K > 1$ to compensate for the fact that a typical glass surface transmits a much larger portion of the incident light than it reflects[5].

Images $\tilde{I}_R$ and $\tilde{I}_T$ can reproduce the types of reflections described above, but are limited to those cases for which $L_R - L_T$ changes smoothly with P_S. However, as shown in Figure 5.2 ❺, the reflection can drop abruptly following the boundaries of an object. This may happen when an object is much closer than the rest of the scene, or when it has a different reflectivity than the surrounding objects. To properly model this behavior, we treat it as a type of reflection on its own, which we apply to a random subset of the image whose range we have already expanded. Specifically, we set to zero the regions of the reflection or transmission layer, whose

[4] Approximating the camera response function with a gamma function does not affect the accuracy of our results, as we are not trying to produce data that is radiometrically accurate with respect to the original scenes.

[5] At an AOI of $\pi/4$, for instance, a glass surface reflects less than 16% of the incident light.

Figure 5.9: Result of our non-rigid deformation model applied to synthetic data. The grid lines are drawn for easier visual inspection and not used during training. We crop an inner patch to avoid impacts from border effects.

intensity is below $T = \text{mean}(\tilde{I}_R + \tilde{I}_T)$, similarly to the method proposed by Fan *et al.* [Fan+17a].

Dealing with Dynamic Scenes

Our approach requires images captured under three different polarization angles. As of now, the standard way to capture different polarization images is sequential and this causes complications for non-static scenes. As mentioned in Section 5.1, if multiple pictures are captured from different locations, the relative motion between the transmitted and reflected layers can help disambiguate them. In our case, "non-static" refers to the scene itself, such as is the case when a tree branch moves between the shots. Several approaches were proposed that can deal with dynamic scenes in the context of stack-based photography. Rather than requiring some pre-processing to fix artifacts due to small scene changes at inference time, however, we propose to synthesize training data that simulates them, such as local, Non-rigid Deformation (NRD). We first define a regular grid over a patch, and then we perturb each one of the grid's anchors by (dx, dy), both sampled from a Gaussian of different variance chosen uniformly from $\{0.1, 0.05, 0.025\}$ for each patch. We then interpolate the position of the rest of the pixels in the patch. For each input patch, we generate three different images, one per polarization angle. We only apply this processing to a subset of the synthesized images — the scene is not always dynamic. For a given patch our method produces distorted patches such as those in Figure 5.9.

Geometry of the Semi-reflective Surface

The images synthesized up to this point can be thought of as the irradiance of the unpolarized light at the semi-reflector. After bouncing off, or going through, the surface, light becomes polarized as described in Section 5.2.1. The effect of a linear polarizer placed in front of the camera and oriented at a given polarization angle, depends on the AOI of the specific light ray. Some previous works assume this angle to be uniform over the image, which is only true if the camera is at infinity,

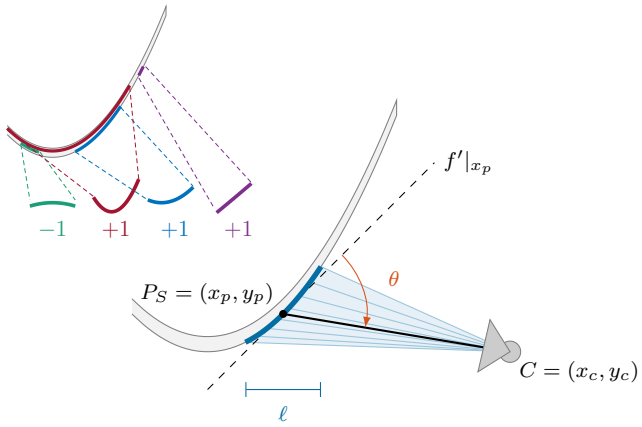


Figure 5.10: Example of our non-rigid motion deformation (right) and a curved surface-generator given the camera position C, a surface-point P_S, length ℓ, and the convexity ± 1 (left).

or if the surface is flat.

We observe that real-world surfaces are hardly ever perfectly flat. Many common glass surfaces are in fact designed to be curved, as is the case of car windows, see Figure 5.3. Even when the surfaces are meant to be flat, the imperfections of the glass manufacturing process introduce Local Curvature Generation (LCG), see Figure 5.2 ❹.

At training time, we could generate unconstrained surface curvatures to account for this observation. However, it would be difficult to sample realistic surfaces. Moreover, the computation of the AOI from the surface curvature may be non-trivial. As a regularizer, we propose to use a parabola. When the patches are synthesized, we just sample four parameters: the camera position C, a point on the surface $P_S = (x_p, y_p)$, a segment length ℓ, and the convexity as ± 1, Figure 5.10. Since the segment is always mapped to the same output size, this parametrization allows generating a number of different, realistic curvatures. Additionally, because we use a parabola, we can quickly compute the AOI in closed form, from the sample parameters. For randomly sampled camera positions (x_c, y_c) the AOI θ is given as

$$\theta = \tan^{-1}\left(\frac{2x_c x_p + x_p^2 - y_c}{4x_c x_p^2 - 4x_p^3 + x_c - x_p}\right). \tag{5.6}$$

Note, that the camera can be placed on both sides of the parabola and different segment lengths give different curvatures.

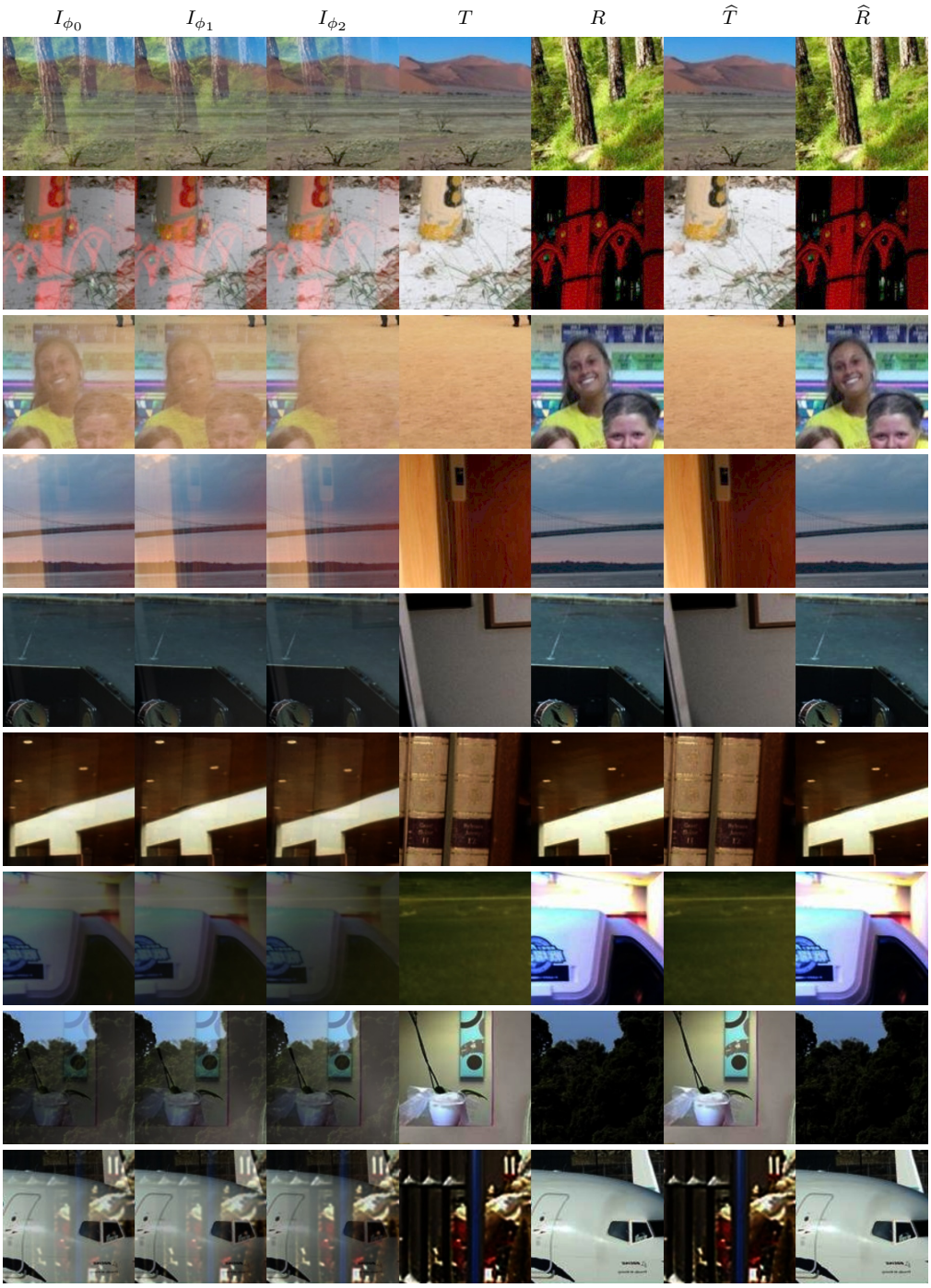


Figure 5.11: Randomly sampled training data from PLACE2 [Zho+17] with synthesized observations I_{ϕ_k} from the ground truth data T and R, and relative estimates $\widehat{T}, \widehat{R}$. (Ground-truth images from [Lin+14].)

5.3 Experiments

In this section we evaluate our method and data modeling pipeline on both synthetic and real data. For the latter, we introduce the Urban Reflection Database (URD), a new dataset of images containing semi-reflectors captured with polarization information. A fair evaluation can only be done against other polarization-based methods, which use multiple images. For the sake of completeness, we also compare against single-image methods for completeness.

The Urban Reflections Dataset (URD). For practical relevance, we compile a dataset of 28 high-resolution RAW images (24MP) that are taken in urban environments using two different consumer cameras (Alpha 6000 and Canon EOS 7D, both ASP-C sensors), and which we made publicly available. This dataset includes examples taken with a wide aperture, and while focusing on the plane of the semi-reflector, thus meeting the assumptions of Fan *et al.* [Fan+17a].

Notes on Previous Methods To perform a thorough evaluation against state-of-the-art methods whose implementation are not publicly available, we re-implemented several representative methods. Here, we give some insights about specific issues with these methods.

The approach of Schechner *et al.* [SSK00] assumes a spatially-invariant AOI θ and seeks to find such a unique local minimum as the best guess for the AOI. We exactly re-produced the experiments of the work by Schechner *et al.* [SSK00]: The exhaustive search is generally able to find the ground-truth AOI in synthetic examples, where the uniform AOI assumption holds (Figure 5.12 left). As already discusses in Section 5.2.1, this assumption rarely holds and fails for real-world examples (Figure 5.12 right). For a fair evaluation, we additionally sample local minima yielding different AOIs, manually inspected the candidate solutions, and cherry-pick the best solution from Schechner *et al.* [SSK00] regarding transmission/reflection result. This method [SSK00] makes the assumption, that $I_\perp, I_\parallel$ are known. Therefore, we follow [KTS] to estimate these projections. Further, to reduce non-rigid motion artifacts, we down-sample the inputs for this method by a factor f_d for estimating the AOI. Empirical evidence suggest $f_d = 16$ as a good choice, which consistently gives a significant performance gain, while reducing the processing time to at most 41 seconds.

Kong *et al.* [KTS] proposed a patch-based extension, which we also re-implemented and accelerated by OpenMP and CUDA. As their proposed method already requires more than five hours computation time for input-size of 256×256px, we compute the AOI at that resolution and then up-sample it for the final layer separation.

Fan *et al.* [Fan+17a] provide a pre-trained neural network. While its architecture fully-convolutional, the large memory footprint limits its application to rather small

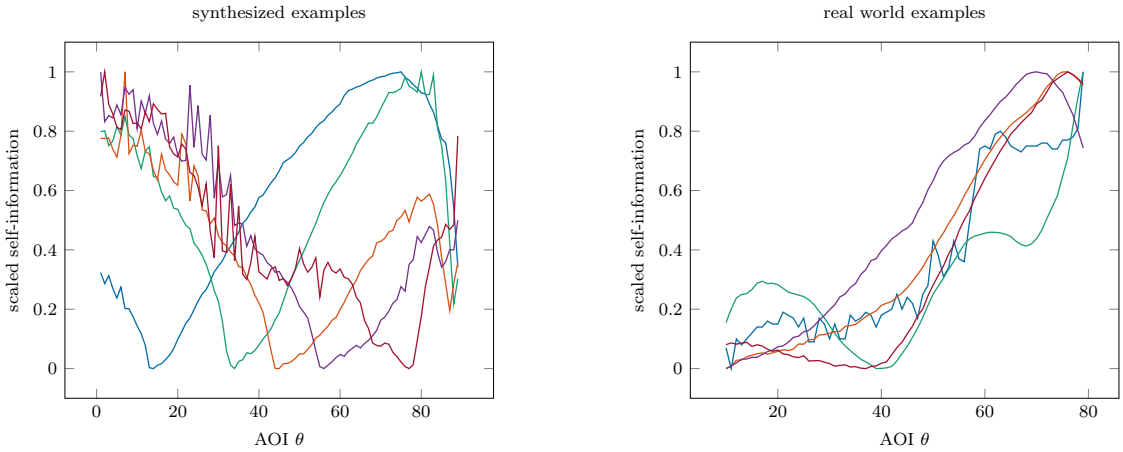


Figure 5.12: Exhaustive search approach of Schechner *et al.* [SSK00] on different examples: On the five synthetic examples produced by the image formation model with uniform AOI the search gives a unique local minimum (left), which corresponds to the perfect solution. This exactly re-produces the experiments done of Schechner *et al.* [SSK00]. However, in most real-world cases, *e.g.*, the five examples on the right, such a local minimum rarely exists and, when it does, it may not represent the best AOI for the layer separation.

images. Therefore, we had to down-sample the single-image input, so that the inference pass would fit the GPU (at most 700 × 500px per image).

For our dataset, we particularly capture an example which meets the requirement of Shih *et al.* [Shi+15] by having ghosting-cues from double-pane windows. The reference implementation by the authors takes 46.12 min for the layer separation process on a down-sampled input image of size 216 × 324px.

Reference implementations provided by the authors exists for the methods [LB14; ADAS17]. We down-sampled the input to meet the memory requirements and to assess a reasonable computation time.

5.3.1 Numerical Performance Evaluation

Due to the need for ground-truth, a large-scale numerical evaluation can only be performed on synthetic data. For this task we take two datasets, the VOC2012 [Eve+] and the PLACE2 [Zho+17] datasets. Comparison with state-of-the-art methods shows that our method outperforms the previously best method by a significant margin in terms of PSNR: ~ 2 dB, see Table 5.1.

For a numerical evaluation on real data, we set up a controlled scene in a lab with a glass surface and objects causing reflections. The camera is placed between the scene and a black background (wall or curtain), which will effectively consti-

Table 5.1: Cross-validation on synthetic data. Best results in bold.

	PASCAL VOC 2012		PLACE2	
Method	RMSE	PSNR	RMSE	PSNR
Farid *et al.* [FA99]	0.401	7.93	0.380	8.38
Kong *et al.* [KTS]	0.160	15.88	0.156	16.12
Schechner *et al.* [SSK00]	0.085	21.34	0.086	21.27
Fan *et al.* [Fan+17a]	0.080	21.89	0.084	21.48
Ours	**0.064**	**23.83**	**0.066**	**23.58**

tute the transmission component by removing unwanted reflection out-side from the controlled scene. Figure 5.13 (right) shows such a setup, where we place a thin glass pane in front of a LEGO figure. Marker pens, were positioned on the camera side causing the wanted reflection. After capturing polarization images of the scene, we removed the marker pens and captured the ground truth transmission. Figure 5.13 shows the transmission images estimated by different methods. Our method achieves the highest Peak signal-to-noise ratio (PSNR) and the least amount of artifacts. The pen markers on the lower right of the images are capture for better comprehension of the scene. The PSNR experiment has been done on a cropped patch without the direct observation.

5.3.2 Effect of Data Modeling

We also thoroughly validate our data-generation pipeline. Using both synthetic and real data, we show that the proposed NRD procedure and the LCG are both effective and necessary. To do this, we train our network until convergence on three types of data: data generated only with the proposed DR, data generated with DR+NRD, and data generated with DR+NRD+LCG. We evaluate these three models on a hold-out synthetic PLACE2 validation set [Zho+17] that features all the transformations from Figure 5.8. The values in Figure 5.14 shows that the PSNR drops significantly when only part of our pipeline is used to train the network. Unfortunately, a numerical evaluation is only possible when the ground truth is available. However, Figure 5.14 shows the output of the three models on the real image from Figure 5.3. The benefits of using the full pipeline are apparent.

As our method can deal with curved surfaces and dynamic scenes, we achieve better performance than the state-of-the-art methods on common scenes, see Figure 5.3.

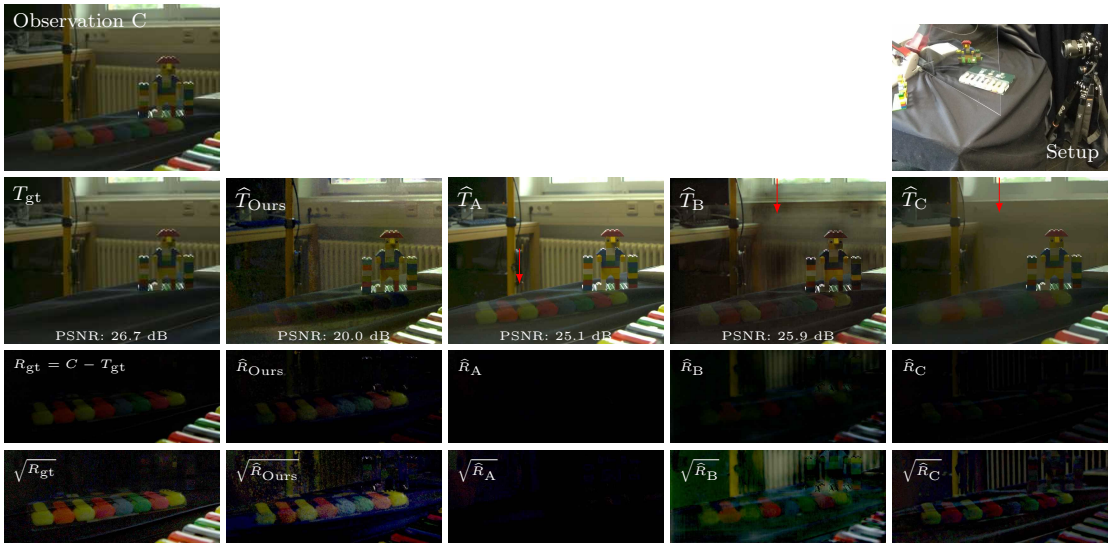
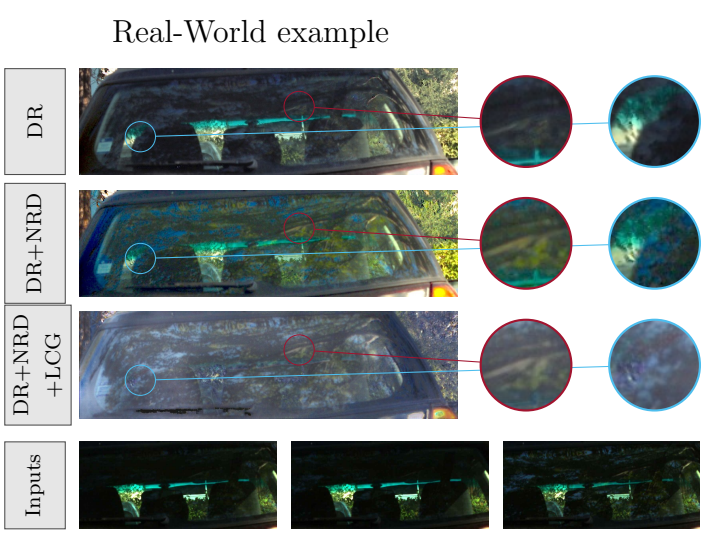


Figure 5.13: By removing the marker pens, we can capture the ground truth transmission, T_{gt}, optically without the problems of attenuated light and possible pollutions of the glass pane. The comparison is done against A: [SSK00], B: [Fan+17a] and C: [ADAS17].

5.3.3 Evaluation on Real-world Examples

We extensively evaluate our method against previous work on the proposed URD. For fairness towards competing methods, which make stronger assumptions or expect different input data, we slightly adapt them, or run them multiple times with different parameters retaining only the best result. Due to space constraints, Figure 5.17 only shows seven of the results. We refer the reader to the Chapter B for the rest of the results in higher resolution. One important remark is in order. Although the images we use include opaque objects, *i.e.* the semi-reflector does not cover the whole picture, the methods against which we compare are local: applying the different algorithms to the whole image and cropping a region is equivalent to applying the same algorithms to the cropped region directly, Figure 5.15.

Figure 5.17, *Curved Window* shows a challenging case in which the AOI is significantly different from θ_B across the whole image, thus limiting the effect of the polarizer in all of the inputs. Moreover, the glass surface is slanted and locally curved, which breaks several of the assumptions of previous works. As a result, other methods completely fail at estimating the reflection layer, the transmission layer, or both. On the contrary, our method separates $\widehat{T}$ and $\widehat{R}$ correctly, with only a slight halo of the reflection in $\widehat{T}$. In particular, notice the contrast of the white painting with the stars, as compared with other methods. While challenging, this scene is far from uncommon.



Synthetic data evaluation
PLACE2 validation

DR	28.17 dB
DR+NRD	30.44 dB
DR+NRD +LCG	31.18 dB

Figure 5.14: Our reflection estimation (left) on a real-world curved surface and synthetic data (right Table) using the same network architecture trained on different components of our data pipeline. Only when using the full pipeline (DR+NRD+LCG) the reflection layer is estimated correctly. Note how faint the reflection is in the inputs (bottom row).

In Figure 5.17, *Bar* one can see another result on which our method performs significantly better than most related works. On this example, the method by Schechner *et al.* [SSK00] produces results comparable to ours. However, recall that, to be fair towards their method, we exhaustively search the parameter space and hand-pick the best result. Another thing to note is that our method may introduce artifacts in a region for which there is little or no information about the reflected or transmitted layer in any of the inputs, such as the case in the region marked with the red square on our $\widehat{T}$. We also show an additional comparison showing the superiority of our method (Figure 5.17, *Paintings*) and a few more challenging cases. Please note, that in a few examples, our method may fail at removing part of the "transmitted" objects from $\widehat{R}$, as is the case in Figure 5.17, *Chairs*.

User Study Since we do not have the ground truth for real data, we evaluate our method against previous results by means of a thorough user study. We asked 43 individuals not involved with the project, to rank our results against the state-of-the-art [FA99; Fan+17a; LB14; SSK00; KTS] and the input images. In our study, we evaluate $\widehat{R}$ and $\widehat{T}$ as two separate tasks, because different methods may perform better on one or the other. For each task, the subjects were shown an input image,

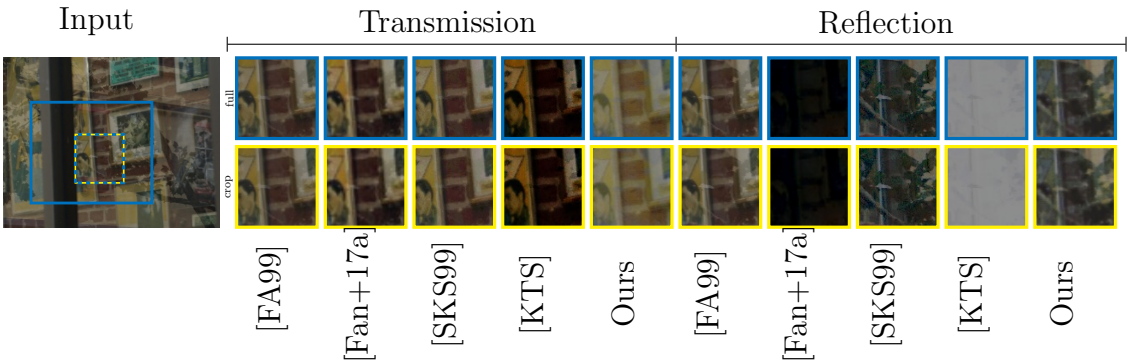


Figure 5.15: Applying different algorithms to the entire image and cropping a region (blue) is equivalent to applying the same algorithms to the cropped region directly (yellow).

Table 5.2: We report the average recall-rate for each method from the user study.

	Transmission		Reflection	
Method	*R*@1	*R*@2	*R*@1	*R*@2
Ours	**0.46**	**0.65**	**0.34**	**0.54**
[SSK00]	0.14	0.38	0.23	0.40
[KTS]	0.11	0.27	0.09	0.20
[Fan+17a]	0.06	0.17	0.08	0.20
[LB14]	0.08	0.21	0.10	0.29
[FA99]	0.06	0.13	0.15	0.37

and the results of each method on the same screen, in randomized order. To ease the task for the participants, we created a custom user interface which provides drag and drop functionality, see Figure 5.16. They were given the task to rank the results from 1 to 6 using drag and drop, which took, on average, 35 minutes per subject. We measure the recall rate in ranking, $R@k$, *i.e.* the fraction of times a method ranks among the top-k results. Table 5.2 reports the recall-rates. Two conclusions emerge from analyzing the table. First, and perhaps expected, polarization-based methods outperform the other methods. Second, our method ranks higher than related works by a significant margin.

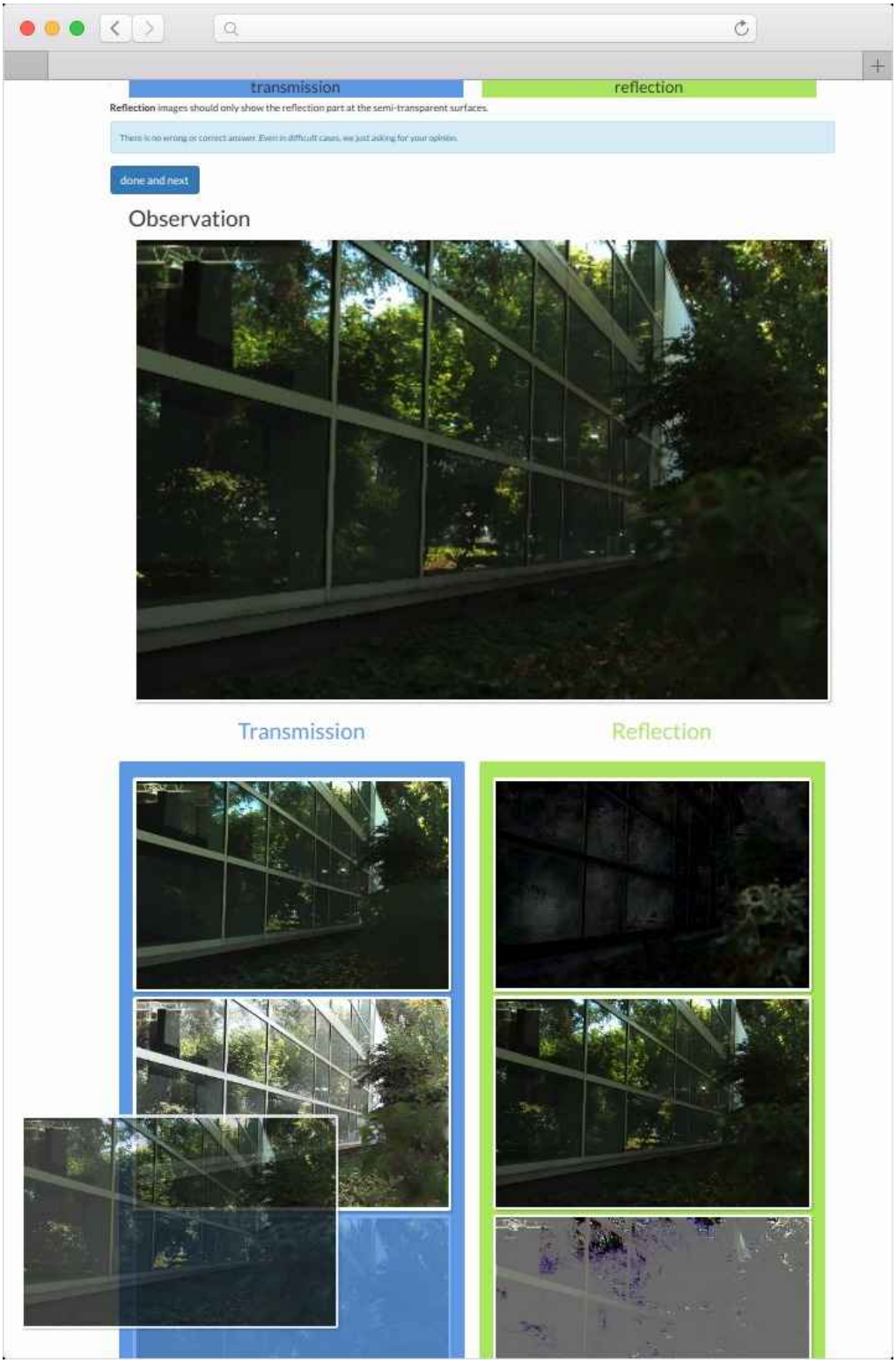


Figure 5.16: Created web-interface for the online user study. Participants ordered the estimated reflections and transmissions using drag and drop.

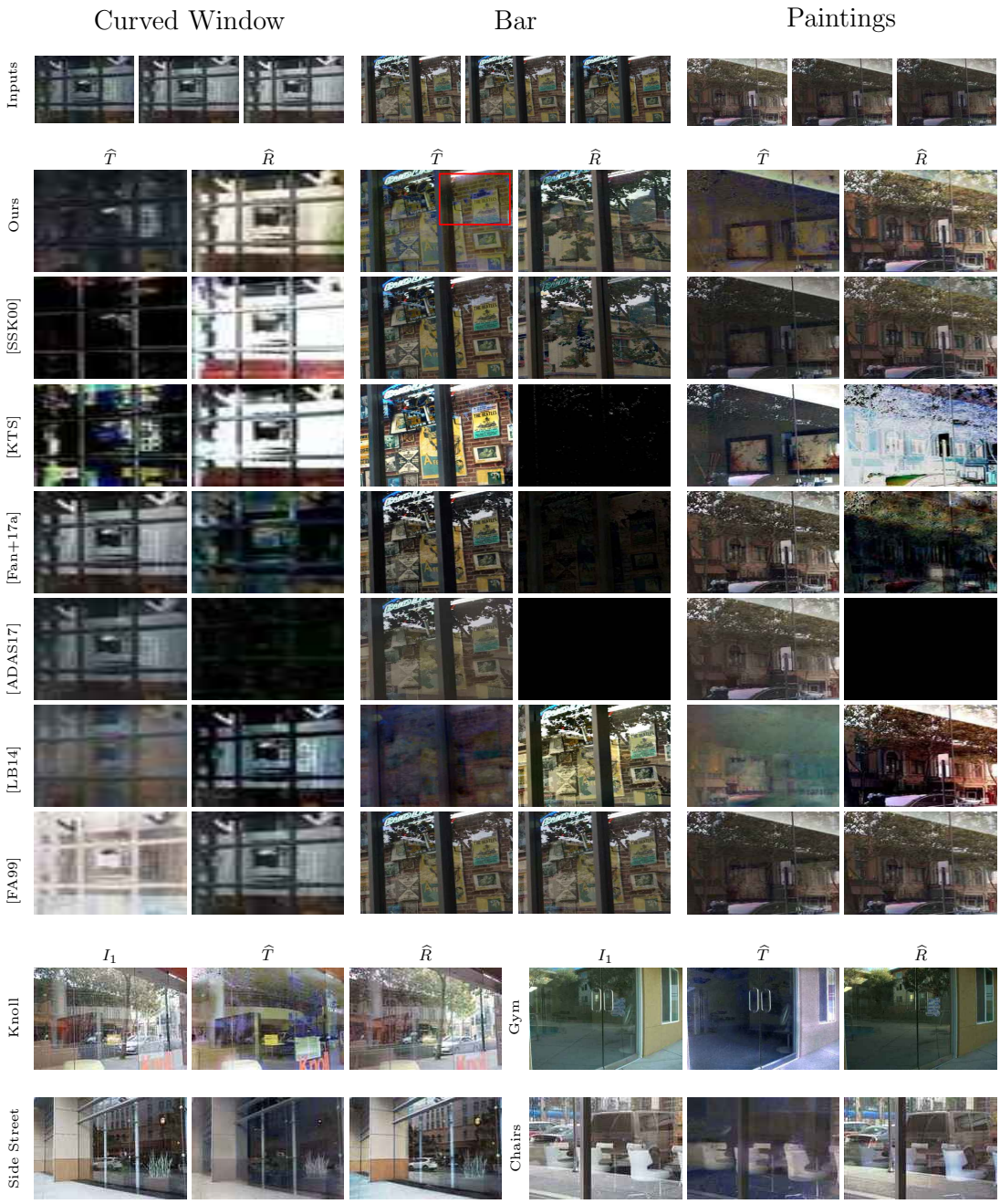


Figure 5.17: Results on typical real-world scenes. Top pane: comparison with state-of-the-art methods, bottom pane: additional results. More results are given in Chapter B.

Rule #22:
GPU memory should be reserved for larger batch sizes and not replace elegant solutions with sheer computing power.

Chapter 6

Multi-Frame Deblurring

As handheld video cameras are now commonplace and available in every smartphone, recording images and videos can be done almost everywhere at any time. However, taking such a quick shot frequently yields a blurry result due to unwanted camera shake during recording or moving objects in the scene. This is especially true in low-light environments. Removing these artifacts from the blurry recordings is a highly ill-posed problem as neither the sharp image nor the motion blur kernel is known. Propagating information between multiple consecutive blurry observations can help restore the desired sharp image or video.

The material of this chapter is based on the following publications:

[Wie+16b] Patrick Wieschollek, Bernhard Schölkopf, Hendrik P. A. Lensch, and Michael Hirsch. „End-to-End Learning for Image Burst Deblurring". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2016. DOI: 10.1007/978-3-319-54190-7_3

[Wie+17] Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, and Hendrik P. A. Lensch. „Learning Blind Motion Deblurring". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017. DOI: 10.1109/ICCV.2017.34

Nowadays, when consumer cameras are built in every smartphone capturing a quick shot is possible with low effort. However, under the absence of tripods (carrying such equipment is the opposite being of portable) such a hand-held acquisition is likely to cause blur when moving camera position during exposure. Let $Y(t_0)$ denote the observed image at time t_0. The final observation Y can thus be written as

$$Y(t_0) = \int_0^{t_0} X(t)dt \tag{6.1}$$

for exposure time t_0 and recording $X(t)$ at t. Hence, when $X(t) \neq X(t')$ for different time steps t, t' the observation becomes a super-imposed combination of different observations introducing artifacts. Consequently, there exists two independent

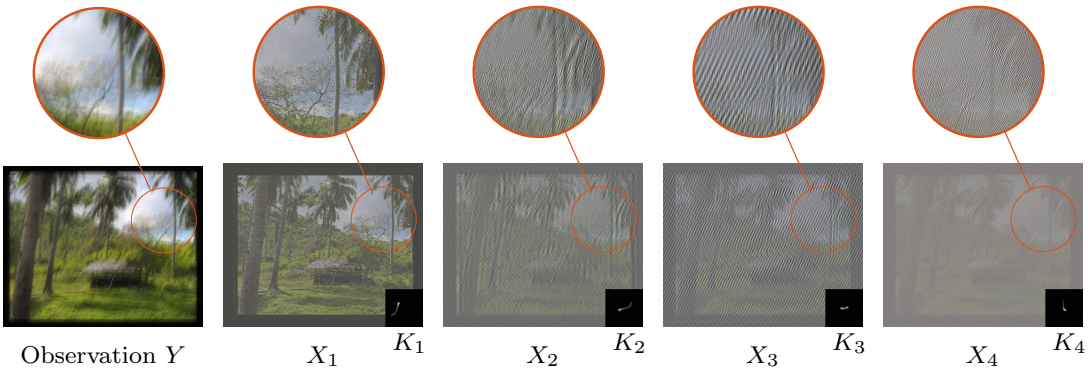


Figure 6.1: A single observations Y can be explained by multiple latent images X_i convolved with an appropriate kernel K_i, *i.e.* $Y = K_i * X_i$ holds for all $i = 1, 2, 3, 4$. The associated blur kernel K_i is illustrated in the bottom right corner.

sources leading to blur: *ego-motion*, *i.e.* moving the camera, and *object-motion*, *i.e.* scene is dynamic.

Static scenes A blurry observation Y caused by ego-motion is usually modeled [KH96] as a spatially invariant convolution of a latent sharp image X with an unknown blur kernel K in

$$Y = K * X + \varepsilon, \tag{6.2}$$

where $*$ denotes the convolution operator and ε models additive zero-mean noise. This model can be rephrased in Fourier space which leads to

$$\mathcal{F}(Z_Y Y) = \mathcal{F}(Z_K K) \odot \mathcal{F}(Z_X X) \tag{6.3}$$

when omitting noise ε, where $\odot$ denotes element-wise multiplication, $\mathcal{F}(J)$ denotes the Fourier-Transformation and Z_J zero-padding of matrix J. Given a particular blur kernel K, recovering X is directly possible via

$$\frac{\mathcal{F}(Z_K K)^* \odot \mathcal{F}(Z_Y Y)}{\mathcal{F}(Z_K K)^* \odot \mathcal{F}(Z_K K)} = \mathcal{F}(Z_X X) \tag{6.4}$$

where z^* denotes the element-wise complex conjugate of z. From Equation (6.4) the ill-posedness of single image blind deconvolution, *i.e.* recovering X from a single observation Y without knowing K, becomes immediately apparent — even without considering noise.

Figure 6.1 illustrates several mathematically valid (K, X) solution-pairs explaining the same blurry observation Y. While (K_1, X_1) seems to be the most likely

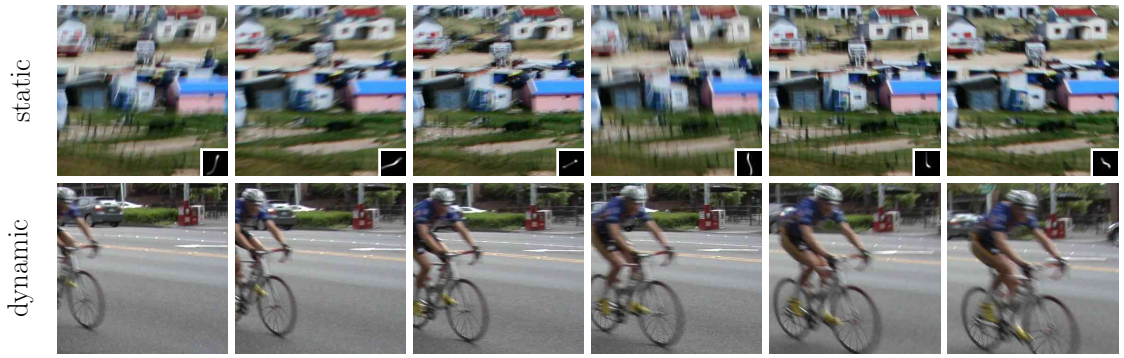


Figure 6.2: For static scenes the blur is solely induced from camera shake (ego-motion, top row). In contrast a cyclist passing by causes motion blur (bottom row) while the background remains relatively sharp. Aligning these frames capturing the dynamic scene is a non-trivial task. (Images from [DS15a; DS15b].)

solution in Figure 6.1, $(K_2, X_2), (K_3, X_3), (K_4, X_4)$ are mathematically feasible solutions as well, indistinguishable in Equation (6.2) from (K_1, X_1).

Fortunately, modern camera sensors allow a fast read-out, such that instead of relying on a single observation Y, the information of multiple observations from an "image-burst" can alleviate the ill-posedness of the problem by using the model

$$Y(t) = K(t) * X + \varepsilon(t), \quad \text{for } t = 1, 2, \ldots, n \tag{6.5}$$

to estimate the latent sharp image X assuming $K(i) \neq K(j)$ for pair-wise distinct $i \neq j$. Still, the problem remains ill-posed. But some valid solutions from the single-image setting as shown in Figure 6.1 could be potentially explained away by considering multiple observations under additional assumptions like static and aligned scene properties.

Dynamic scenes The handling of multi-frame inputs requires a spatial alignment of successive observations. While aligning images of static scenes can be solved by estimating a homography between different observations, in dynamic scenes image-parts might change and therefore hinder a proper alignment in addition to violating the assumption $X = X(t)$ made in Equation (6.5), see Figure 6.2. The relative movement of the cyclist causes blur, even if the background features relatively low blur. Hence, the model changes to

$$Y(t) = K(t) * X(t) + \varepsilon(t), \quad \text{for } t = 1, 2, \ldots, n \tag{6.6}$$

where $X(t)$ and $X(t+1)$ share similar content but scene parts are eventually shifted over time t.

6.1 Related Work

The problem of image deblurring can be formulated as a *non-blind* or a *blind* deconvolution version, depending on whether information about the blur kernel K is available or not. Blind image deblurring (BD) is quite common in real-world applications and has seen considerable progress in the last decade. A comprehensive review is provided in the overview article by Wang and Tao [WT14].

Due to the ill-posed nature of this problem (see Figure 6.1), traditional state-of-the-art methods such as Sun *et al.* [Sun+13] or Michaeli and Irani [MI14] use carefully chosen patch-based priors for sharp image prediction to identify a meaningful X amongst all possible solutions. Besides the deblurring images under stationary blur [Lev+09] assumption, Köhler *et al.* [Köh+12] recorded real camera motion for the case of non-uniform blur.

Data-driven methods based on neural networks have demonstrated success in non-blind restoration tasks [Sch+13; Xu+14; RW15] as well as for the more challenging task of BD where the blur kernel is unknown [Sch+15; Sun+15; Cha16; Hra+15; Svo+16]. Using a large corpus of training examples they directly learn an image-prior to estimate a natural image X during the reconstruction. Removing the blur from moving objects from a single observation has been recently addressed [NCF17].

To alleviate the ill-posedness of the problem [Has+09], one might take multiple observations into account. Hereby, observations of a static scene, each of which is differently blurred, serve as inputs [ZWZ14; Che+08; Cai+09; ŠM12; ZŠM12; Hir+10]. To incorporate video properties such as temporal consistency previous methods [ZC14; ZY15; KNL16; Ito+14] use powerful and flexible generative models to explicitly estimate the unknown blur along with predicting the latent sharp image. However, this comes at the price of higher computation cost, which typically requires tens of minutes for the restoration process for a few low-resolution frames.

To accomplish faster processing times Delbracio and Sapiro [DS15b] have presented a smart way to average a sequence of input frames based on Lucky Imaging methods. Instead of an align-and-average approach in the spatial domain, they propose to compute a weighted combination of all aligned input frames in the Fourier domain which favors stable Fourier coefficients in the burst containing sharp information. This Fourier-Burst-Accumulation (FBA) approach yields much faster processing times and removes the requirement to compute the blur kernel explicitly. Although removing an explicit deconvolution-step during reconstruction delivers a speed-up it is limited to sequences, which contain at least one reasonable sharp shot.

Independently to our approach, the related task of deblurring of entire video sequences has been later addressed by Su *et al.* [Su+17]. Their approach uses the commonly used U-Net architecture [RFB15] with skip connection to directly regress the sharp image from an input burst similar to denoising approaches [MSY16b].

Their fully convolutional neural network learns an average of multiple inputs with reasonable performance. Unfortunately, this requires to fix the temporal input size at training time and hence limits the amount of usable information, *e.g.* training on sequences of 5 images does not allow to incorporate information from 10 images during inference.

6.2 Learning to Deblur Static Scenes

This section proposes the first discriminative data-driven approach for solving the blind multi-frame deconvolution problem assuming a static scene and aligned input frames. We begin by describing the generation of synthetically blurred images Y given sharp groundtruth data X before discussing an appropriate neural network architecture to estimate $\hat{X}$ given only Y.

6.2.1 Training Data Generation

Following the model from Equation (6.5) the blurry observation can be directly synthesized given a corpus of sharp ground-truth images $X \in \mathcal{X}$ and sampled blur kernels $K_i(t) \in \mathcal{K}$. We randomly sample image-patches from the MS COCO dataset [Lin+14], which contains real-world photographs collected from the internet. To ensure a high quality of ground-truth patches X, we reject potential ground-truth images with too small image gradients, which indicates that the content is likely to either consist of either entirely homogeneous regions without any traits of the blur, or already contains strong blur and therefore should be omitted. The remaining 542217 sharp patches are then blurred on-the-fly using blur kernels sampled from a Gaussian process following the idea of Schuler *et al.* [Sch+15]. Figure 6.3 depicts a random subset of some generated blur kernels and Figure 6.4 illustrates a generated image burst of 8 blurry observations given a sharp image. To obtain robustness against different blur strength, we generate blur kernels of sizes 17×17 and 7×7 pixels. In addition, we apply standard data augmentation methods like rotating and mirroring to the ground-truth data. Hence, this approach gives nearly an infinite amount of training data due to random Point-Spread Function (PSF) kernels. To simulate read-out noise, we add white Gaussian noise with variance $\sigma^2 = 0.1$. The validation data from the official split [Lin+14] is once precomputed to ensure fair evaluation during training.

6.2.2 Network Architecture Design

The network π_θ operates on a patch-by-patch basis. For a burst of observed images $Y(t)$, $t = 1, 2, \ldots, n$, it splits each of these observations into overlapping patches of size 65×65 pixels and the network predicts sharp intermediate patches of size

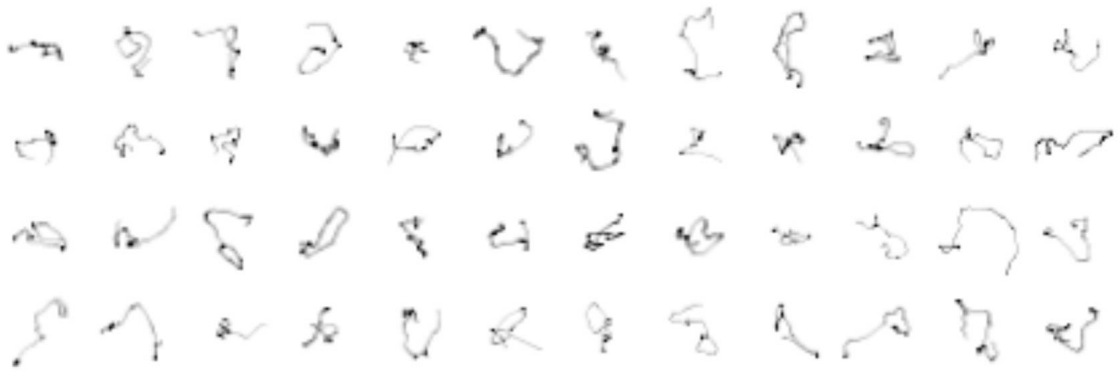


Figure 6.3: Synthetically generated blur kernels $K \in \mathcal{K}$ using 3D Gaussian process, which result is centered. Rescaled for the purpose of illustration.

Figure 6.4: Applying different synthetic blur kernels to the same image (right) gives different aligned observations (left) featuring different blur artifacts.

33×33. All predicted patches are recomposed to form the final prediction $\hat{X}$ by averaging the predicted pixel values and fusing the burst using an extended version of FBA [DS15b]. During training, we optimized the network parameters θ by minimizing the squared ℓ_2 objective

$$\|\pi_\theta\left[Y(1), Y(2), \ldots, Y(n)\right] - X\|_2^2. \tag{6.7}$$

In the following, we will describe the construction of π_θ, the optimization of network parameters θ during the training of the neural network and the restoration of an entire sharp image. The architecture π_θ chains several stages:

(a) Frequency-Band-Analysis with Fourier coefficient prediction,
(b) a deconvolution part and
(c) image fusion.

Figure 6.5 illustrates the first two stages of our proposed system.

(a) Frequency-Band-Analysis. The frequency band analysis computes the discrete Fourier transform of the observed patch $Y(t)$ following the neural network

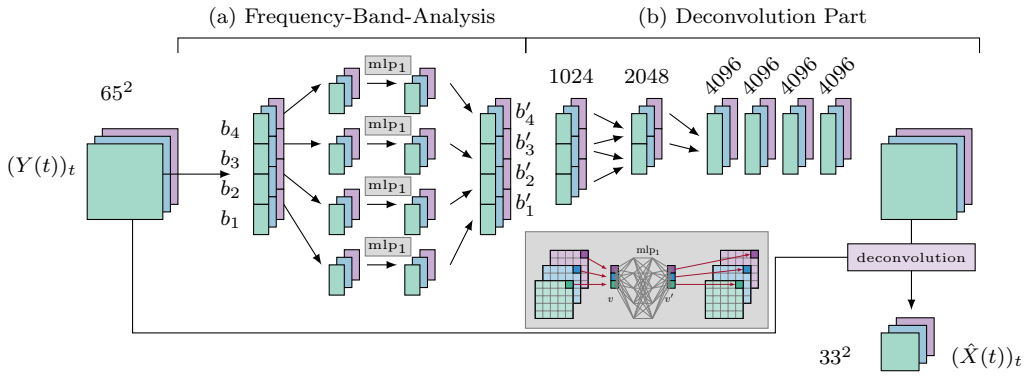


Figure 6.5: Frequency band analysis and deconvolution for an image burst with 3 patches $Y(1), Y(2), Y(3)$. Following the work of Chakrabarti [Cha16] we separate the Fourier spectrum in 4 different bands. In addition, we allow each band separately to interact across all images in one burst to support early information sharing. The predicted output of the deconvolution step are smaller patches $\hat{X}(1), \hat{X}(2), \hat{X}(3)$.

approach in by Chakrabarti [Cha16] at three different sizes ($17 \times 17, 33 \times 33, 65 \times 65$) using different sample sizes, which we will refer to bands b_1, b_2, b_3. In addition, band b_4 represents a low-pass band containing all coefficient $\max |z| \leq 4$ from band b_3. This is depicted in Figure 6.5. To enable early information sharing within one burst of patches, we allow the neural network to spread the per band information extracted from one patch across all images of the burst using 1×1 convolution.

More precisely, the values of one Fourier coefficient $(f_{ij})_t$ at frequency position (i, j) across the entire burst $t = 1, 2, \ldots, n$ can be considered as a single vector $(f_{ij})_{t=1,2,\ldots,n}$ of dimension n (compare $\texttt{mlp}_1$ in Figure 6.5). A transformed version of this excerpt will be placed at the same location in the output patch again. This allows the neural network to adjust the extracted Fourier coefficients right before a dimensionality reduction occurs based on the entire image burst rather than for each observation individually. These modified values $(f'_{ij})_{t=1,2,\ldots,n}$ give rise to adjusted Fourier bands b'_1, b'_2, b'_3, b'_4.

(b) Deconvolution. Pairwise merging of the resulting bands b'_1, b'_2, b'_3, b'_4 with modified Fourier coefficients using fully connected layers with ReLU activation units entails a dimension reduction. The produced 4096 feature vector encoding is then fed through several fully connected layers producing a 4225 dimensional prediction of the filter coefficients of the deconvolution kernel. This essentially resembles a convolution layer in the Fourier space with large spatial support. Applying the deconvolution kernel predicts a sharp intermediate patch $\hat{X}$ of size 33×33 from each input burst. This step is implemented as a multiplication of the predicted Wiener Filter with the Fourier transform of the input patch, similar to Equation (6.4).

(c) Image fusion. In the last part of our pipeline, we fuse all available patches

$Y(1), Y(2), \ldots, Y(N)$ by adopting the FBA approach [DS15a] as a neural network component with learnable weights. The vanilla FBA algorithm applies the following weighted sum to a Fourier transform $\hat{\alpha}$ of the patch α:

$$u(\hat{\alpha}) = \mathcal{F}^{-1}\left(\sum_{i=1}^{k} w_i(\zeta)\hat{\alpha}_i(\zeta)\right)(x) \tag{6.8}$$

$$w_i(\zeta) = \frac{|\hat{\alpha}_i(\zeta)|^p}{\sum_{j=1}^{k} |\hat{\alpha}_j(\zeta)|^p}, \tag{6.9}$$

where w_i denotes the contribution of frequency ζ of a patch α_i. The term $u(\hat{\alpha})$ is differentiable w.r.t. α allowing to pass gradient information to previous layers through back-propagation. The motivation is that Fourier coefficients with stable magnitude $|\hat{\alpha}_i(\zeta)|$ across the entire burst are carrying information about the sharp content, while changing magnitudes indicates blurry information and consequently get a smaller weight. To incorporate this algorithm as a neural network layer into our pipeline, we replace Equation (6.8) by a parametrized version

$$u(\hat{\alpha}) = \mathcal{F}^{-1}\left(\sum_{i=1}^{k} h_\phi(\zeta)\hat{\alpha}(\zeta)\right)(x). \tag{6.10}$$

Hence, instead of a hard-coded weight-averaging (using hand-crafted weights w_i) the network is able to learn a data-dependent weighted-averaging scheme. Again, the function $h_\phi(\cdot)$ represents two 1x1 convolutional layers with trainable parameters ϕ following the same idea of considering the Fourier coefficient across one burst as a single vector (compare $\texttt{mlp}_1$ in Figure 6.5).

6.2.3 Correcting Colors during Inference

During inference, we feed input patches of size 65×65 into our neural network with stride 5. Using overlapping patches helps to average multiple predictions. For the recombination of overlapping patches, we weight each patch content with a two-dimensional Hanning window to favor pixel values in the middle of the patch and devalue information at the edge of the patch.

While the predicted images $\hat{X}$ generated by our neural network contain well-defined sharp edges we observed desaturation in color contrast. To correct the color of the predicted image, we replace its *ab*-channel in the CIE-*Lab* color space by the *ab*-channel of the FBA results (compare Figure 6.6) obtaining our final prediction $\tilde{X}$.

Regarding runtime, the most expensive step is the frequency band analysis. Given a burst of 14 images of size 1000×700 pixels the entire reconstruction process takes roughly 5 minutes per channel with our unoptimized implementation executed partly on the CPU.

Figure 6.6: Deblurring a burst of degraded images from a groundtruth image (left) results in a desaturated image $\hat{X}$ (middle). Therefore we correct those colors $\tilde{X}$ (right) using non-parametric color transfer by histogram-matching. (Inputs images from [Lin+14].)

6.2.4 Experiments

To evaluate and validate our approach we conduct several experiments including a comprehensive comparison with state-of-the-art techniques on a real-world benchmark dataset [DS15a], and performance evaluation on a synthetic dataset to test the robustness of our approach with varying image quality of the input sequence.

Training Details

Unfortunately, sophisticated stepsize heuristics like Adam [KB14b] or Adagrad [DHS11] failed to guarantee a stable training. We suspect the large range of values in the Fourier space to mislead those heuristics based on statistics about exponential moving averages. Instead, we use stochastic gradient descent with momentum ($\beta = 0.9$), batch size 32 and an initial learning rate of $\eta = 2$ which decreases every 5000 steps by a factor of 0.8. Training the neural network took six days using TensorFlow [Aba+15] on a NVIDIA Titan X.

Baseline Comparison

A baseline to our approach is stacking the method of Chakrabarti [Cha16] and Delbracio and Sapiro [DS15a] subsequently, each in a separate step. Therefore, we fine-tuned the provided weights from [Cha16] in combination with our FBA-layer. Figure 6.7 shows the training progress for an exemplary patch, where the improvement in sharpness is clearly visible over different training epochs i.

In addition, we run the entire pipeline of Chakrabarti [Cha16] including the computational demanding non-blind deconvolution EPLL [ZW11] step and afterwards FBA [DS15a]. The ad-hoc combination is significantly slower and results in reconstructions having ringing-artifacts[1] (see Figure 6.8). They are significantly dampened in our joint training approach.

[1]Please refer to Chapter A for more details about such artifacts.

Figure 6.7: Estimates of a single patche over different training epochs (top left to bottom right) compared to the unobserved ground-truth patch ("gt"). Note how the sharpness continuously increases with training.

Figure 6.8: Comparison to a baseline approach of simply stacking multiple approaches [Cha16]+[ZW11]+[DS15a]. An end-to-end training (ours) avoids producing ringing-artifacts which are clearly visible on the roof parts and edges without joint training. (Inputs images from [DS15a].)

Comparison on real-world images

We compare the restored images with other state-of-the-art multi-image blind deconvolution algorithms. In particular, we compare with the multichannel blind deconvolution method from Šroubek et al. [ŠM12], the sparse-prior method of [ZWZ13] and the FBA method [DS15a]. We used the data provided by [DS15a], which contains typical photographs captured with hand-held cameras (iPad back camera, Canon 400D) that contain complex structures in the images. As they are captured under various challenging lighting conditions they exhibit both noise and saturated pixels.

The FBA algorithm [DS15a] demonstrated superior performance compared to previous state-of-the-art multi-image blind deconvolution algorithms [ŠM12; ZWZ13] in both reconstruction quality and runtime. Figure 6.9 shows crops of the deblurred results on these images. Our trained neural network featuring the FBA-like averaging yields comparable if not superior results compared to previous approaches [ŠM12; ZWZ13; DS15a]. In direct comparison to the FBA results (see Figure 6.9), our method is better removing blur due to our additional prepended deconvolution module.

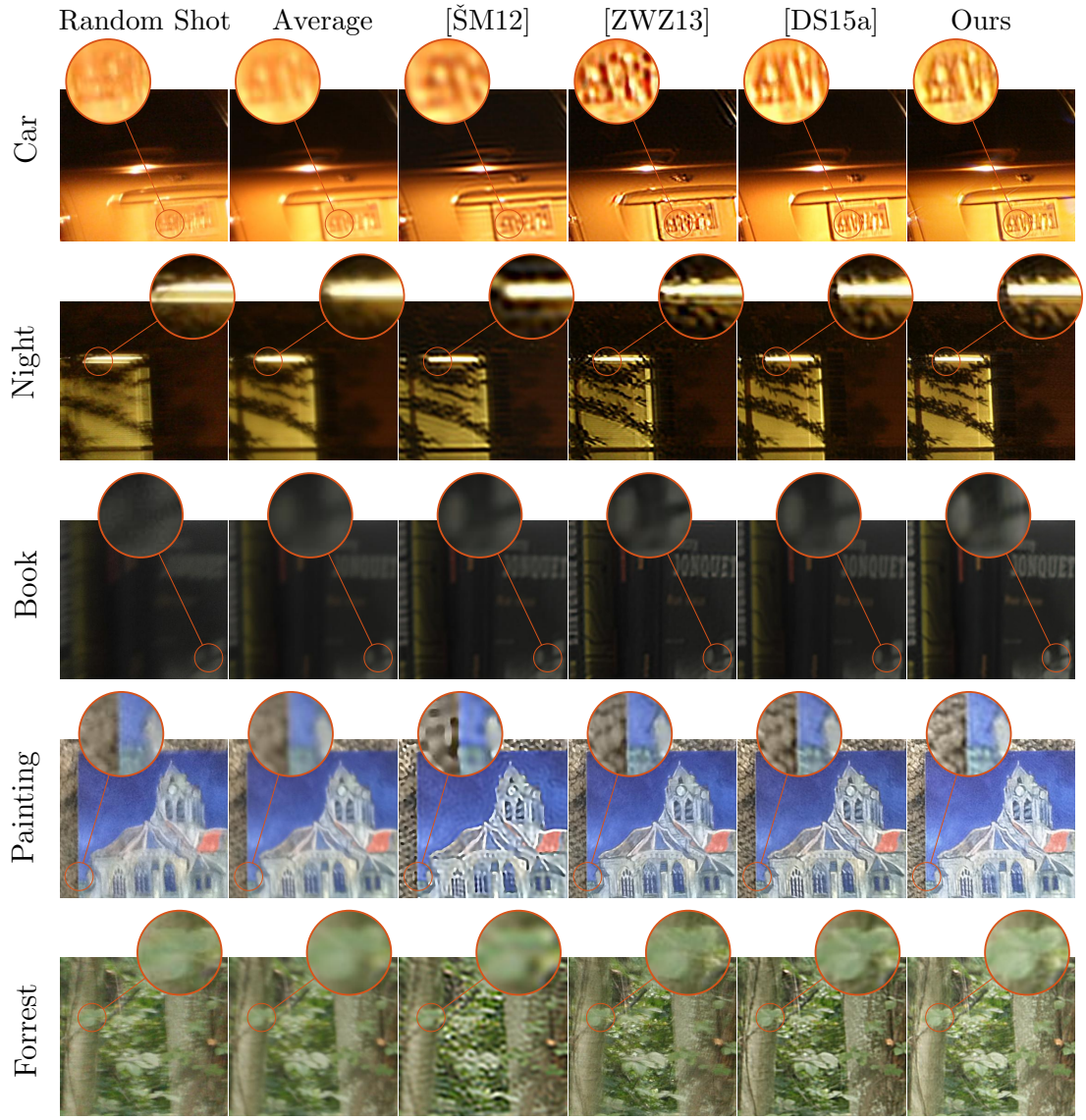


Figure 6.9: Comparison to state-of-the-art multi-frame blind deconvolution algorithms on a real-world benchmark data set. Our approach produces the sharpest results except for the last scene, which could be caused by the color transfer described in Section 6.2.3. (Inputs images from [DS15a].)

6.2.5 Deblurring Bursts with varying Number of Frames and Quality

A burst of images might share different levels of blurriness potentially featuring a relatively sharp shot being handy for Lucky Imaging methods, like FBA [DS15b].

To analyze the performance of our approach depending on the burst "quality", we sorted all images provided by [DS15a] within one burst according to their PSNR beginning with images of strong blur and consequently adding sharper shots to the burst gives a series of bursts starting with images of poor quality up to bursts with at least one close-to-sharp shot. Since our architecture is trained for deblurring bursts with exactly 14 input images, we duplicated input observations of bursts with fewer frames. Figure 6.10 clearly indicates good performance of our neural network even for a relative small number of input images with strong blur artifacts. For example, the license plate becomes readable using only two blurry observations in our approach, see Figure 6.10 (a). In the book example (b) using only one shot is already sufficient to read the title of the book, while FBA requires at least four input images. The neural network produces consistently sharper outputs (c), and the two most blurry observations are sufficient for our method to significantly sharpen the scene (d), while FBA requires at least six blurry observations to produce a reasonable sharp output. Already capturing two observations which are perfectly aligned without any moving objects is arduous and not ubiquitous suitable. Hence, an approach like ours reducing the number of used observations is beneficial for broader applicability.

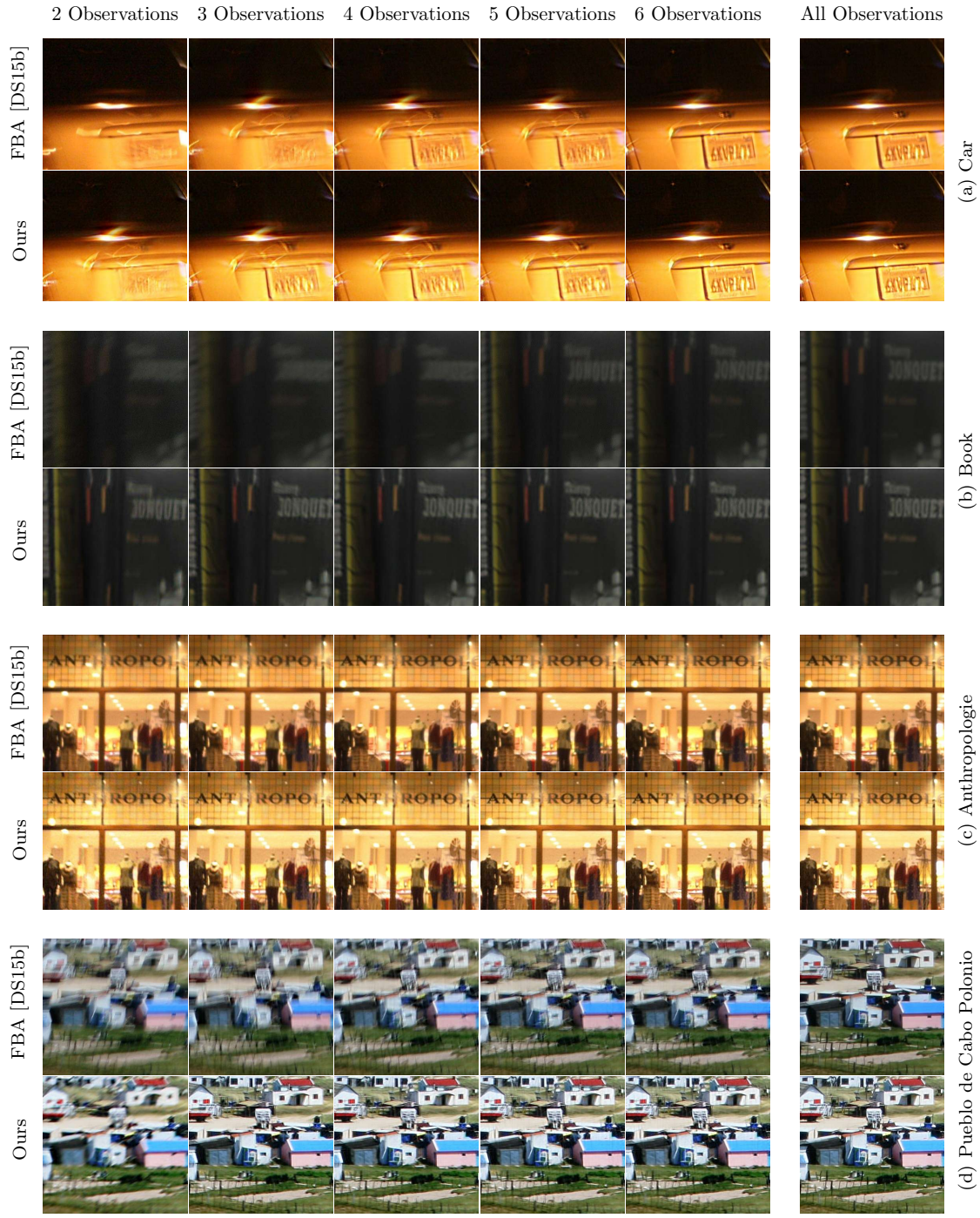


Figure 6.10: Comparison against previous state-of-the-art results limiting the amount of inputs. Our discriminative approach (FourierNet) gets along with fewer observations creating superior sharp reconstructions. (Inputs images from [DS15a].)

6.3 Learning to Deblur Dynamic Scenes

While generating blurry observations of static scenes from the previous section is based on evaluating Equation (6.2), synthesizing blur caused by object motion for training data is notoriously difficult. It requires realistic training data with two coordinated versions for each frame: a blurred version as input and an associated sharp version as ground truth. Obtaining real-world ground-truth data with dynamic scenes itself is a challenge, as any recorded sequence could suffer from the described blur effects. In addition, such a generation mechanism needs to scale to several thousand training points when optimizing a convolutional neural network for the task of deblurring dynamic scenes. In the following section, we will discuss possible methods to collect such reliable ground-truth data.

6.3.1 Synthesizing Dynamic Scenes for Training

Recent works [Su+17; NCF17] have built a training data set by manually recording videos captured at 240fps with a GoPro Hero camera to minimize the blur in the ground-truth. Frames from these high-fps videos are then processed and averaged to produce plausible motion blur synthetically. While they described significant effort to capture a broad range of different situations, this process is limited in the number of recorded samples, in the variety of scenes and the used recording devices. For fast moving objects, artifacts are likely to arise due to the finite framerate. In addition, such a manual recording at high fps is prone to low signal-to-noise ratio using short exposure times effectively limited to outdoor scenes. Both disadvantages come to bear when using even more professional equipment like the Fastec TS5Q camera as done by Janai *et al.* [Jan+17] for optical flow estimation.

We also tested this method for generating training data and inspected the data recorded by Janai *et al.* [Jan+17], but found it hard to produce a large enough and diverse dataset of sharp ground-truth videos of high quality. An alternative way is to render synthetic scenes as shown in Figure 6.11, *e.g.* using Blender [Ble17]. While this approach can accurately incorporate both sources of blur, it is limited by the number of available synthetic scenes and demanding rendering time, as it demands a higher amount number of samples per pixel during rendering.

Rather than acquiring training data manually or synthesizing it using 3D rendering software, we propose to acquire and filter video data from online media. As people love to share and rate multimedia content, each year millions of video clips are uploaded to online platforms like YouTube. The video content ranges from short clips to professional videos of up to 8k resolution. From this source, we have collected videos with 4k-8k resolution and a frame rate of 60fps or 30fps. The video content ranges from movie trailers, sports events, advertisements to videos on everyday life. To remove compression artifacts and to obtain slightly sharper ground-truth we resized all collected videos by the factor 0.25 respectively 0.125,

Ego Motion (Observation) | Ego Motion (Ground-truth) | Object Motion (Observation) | Object Motion (Ground-truth)

Figure 6.11: Synthetic scenes consisting of a planar background photo and synthetic 3D object in the foreground. A rendering engine like Blender [Ble17] can produce physically correct blurry observation and ground-truth pair. Rendering for each pair takes 14 seconds using Blender [Ble17].

finally obtaining full-HD resolution videos.

Consider such an acquired video with frames $(\mathbf{F_t})_{t=1,2,\ldots,T}$. For each consecutive frame pair $(\mathbf{F_t}, \mathbf{F_{t+1}})$ at time t we compute n additional synthetic sub-frames between the original frames $\mathbf{F_t}, \mathbf{F_{t+1}}$ resulting in a high frame rate video

$$(\ldots, F_{t-1}^{(n-1)}, F_{t-1}^{(n)}, \mathbf{F_t}, F_t^{(1)}, F_t^{(2)}, \ldots, F_t^{(n-1)}, F_t^{(n)}, \mathbf{F_{t+1}}, F_{t+1}^{(1)}, F_{t+1}^{(2)}, \ldots). \tag{6.11}$$

All sub-frames are computed by blending between the neighboring original frames $\mathbf{F_t}$ and $\mathbf{F_{t+1}}$ warping both frames using the optical flow in both directions $w_{\mathbf{F_t}\to\mathbf{F_{t+1}}}$ and $w_{\mathbf{F_{t+1}}\to\mathbf{F_t}}$. Given both optical flow fields [HS80], we can synthesize an arbitrary number of subframes, see Figure 6.12. For practical purposes, we set $n = 40$, thus implying an effective framerate of more than 1000fps without suffering from low signal-to-noise ratio (SNR) due to short exposure times.

These generated sub-frames are then averaged to synthesize a plausible blurry version

$$\mathbf{B_t} = \frac{1}{1+2L}\left(\mathbf{F_t} + \sum_{\ell=1}^{L} F_t^{(\ell)} + F_{t+1}^{(n-\ell)}\right) \tag{6.12}$$

for each sharp frame $\mathbf{F_t}$. We choose $L \in \{20, 40\}$ to create different levels of motion blur. The entire computation can be done offline on a GPU. For all video frames (5.43 hours in total) that passed our sharpness test (see Section 6.2.1) we produce a ground-truth video clip and blurry version both at 30fps in full-HD. Besides the unlimited amount of training data, another principal advantage of this method is that it incorporates different capturing devices naturally. Furthermore, the enormous amount of video content available allows us to conservatively tune all thresholds and parameters of the pre-processing pipeline to reject low-quality video (too dark, too static) without affecting the overall effective size of the training data. Though the recovered optical flow using the Farneback method is not perfect,

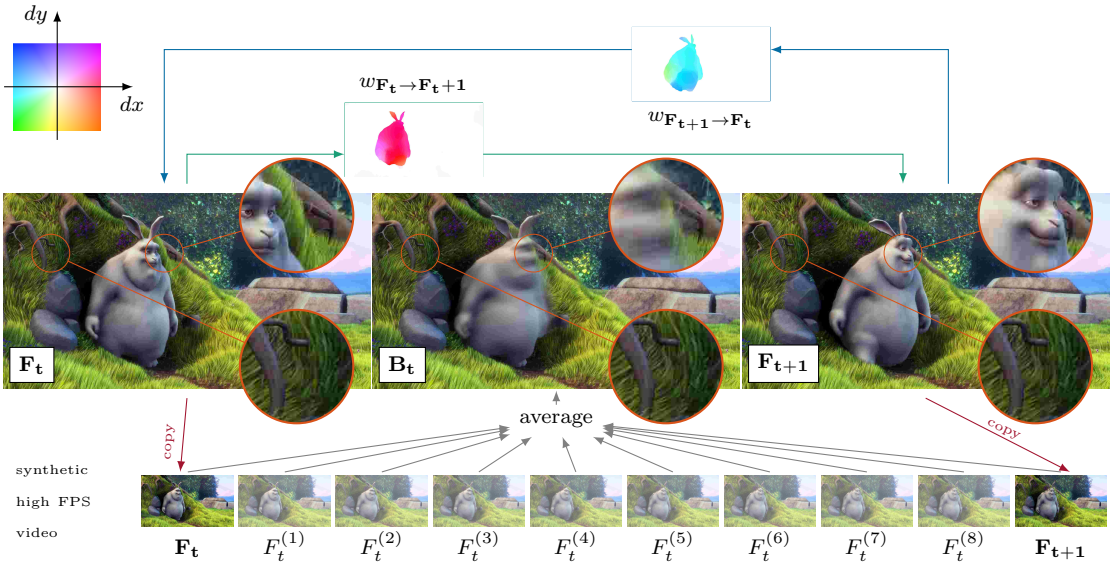


Figure 6.12: For any two consecutive and sharp frames $\mathbf{F_t}, \mathbf{F_{t+1}}$ the estimated optical flow between them can be used to synthesize a high-framerate video as in Equation (6.11) which frames are averaged to simulate motion blur resulting in a blurry frame $\mathbf{B_t}$. The background sharpness is unaffected, while moving objects feature strong blur. (Inputs images from BigBuckBunny.)

we observed an acceptable quality of the synthetically motion blurred dataset. Importantly, tracking pixel movements when computing the optical flow requires reasonable sharp frames, those frames which already passed the sharpness test. Quite recent methods like FlowNet2 [Ilg+17] are now on par regarding the speed with the used optical method but were not available at the time when generating training data. We used FlowNet2 for the illustration in Figure 6.12. To add variety to the training data, we crop random parts from the frames and resize them to 128×128px. Figure 6.13 contains the final blurry training data of 5 consecutive frames with synthesized blur from motion and camera shake along with the ground-truth frames.

6.3.2 Handling the Time Dimension

The typical input shape required by CNNs in computer vision tasks is $[B, H, W, C]$ — batch size, height, width and number of channels. However, processing series of images includes a new dimension: time T. To apply spatial convolution layers the additional dimension has to be "merged" either into the channel $[B, H, W, C \cdot T]$ or batch dimension $[B \cdot T, H, W, C]$. Previous methods [Su+17; Wie+16c] stack the time along the channel dimension rendering all information across the entire

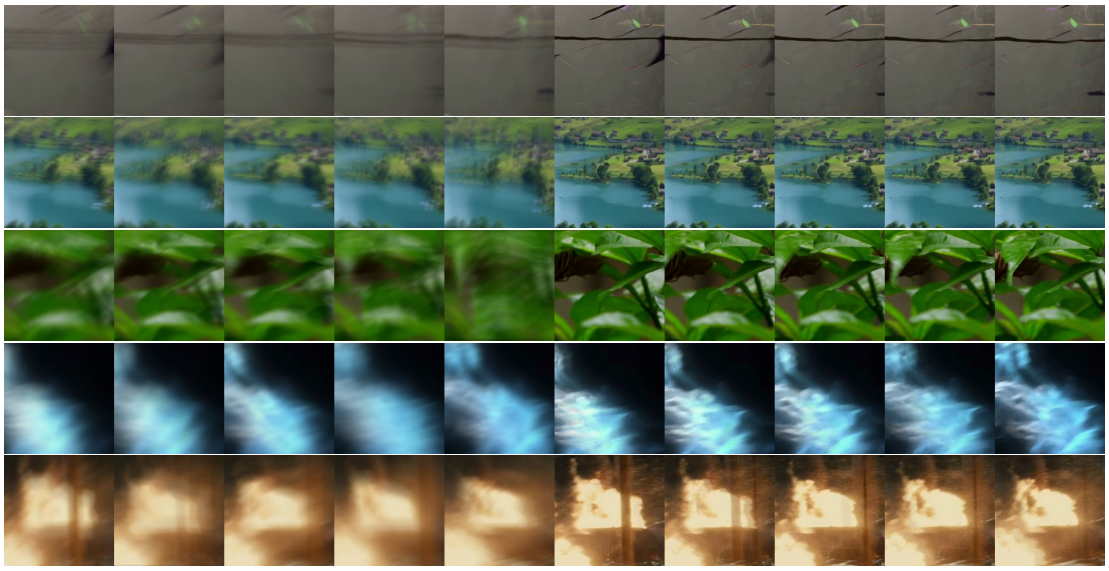


Figure 6.13: Synthesized training data with motion blur and blur from camera shake (left 5 frames, $\mathbf{B_t}$) and corresponding sharp ground-truth frames (right 5 frames, $\mathbf{F_t}$) for $t = 0, 1, 2, 3, 4$.

burst available without further modification. This comes at the price of removing information about the temporal order. Further, the number of input frames needs to be fixed before training, which limits their application. Longer sequences could only be processed with workarounds like padding and sliding window processing. On the other hand, merging the time-dimension into the batch dimension would give flexibility at processing different length of sequences. However, the processing of each frame is then entirely decoupled from its adjacent frames — no information is propagated. Architectures using convLSTM [PHC16] or convGRU cells [Cho+14] are designed to naturally handle time series but they would require several tricks [Lau+16; Wan+13] during training. We tried several architectures based on these recurrent cells but found them difficult to train and observed hardly any improvement even after two days of multi-GPU training.

6.3.3 Recurrent Network Architecture Design

Instead of including recurrent layers, we propose to formulate the entire network as a recurrent application of deblur blocks and successively process pairs of inputs (target frame and additional observation), which gives us the flexibility to handle arbitrary sequence lengths and enables information fusion inside the *deblur block* network.

Consider a single deblur step with the current prediction $\hat{\mathcal{I}}^{k-1}$ of shape $[H, W, C]$. A blurry observation $\mathcal{I}_{-k}$ for $k > 1$ is used to enhance the current prediction $\hat{\mathcal{I}}^{k-1}$ resulting in $\hat{\mathcal{I}}^{k}$, which again is fed along with a new observation $\mathcal{I}_{-k'}, k' > k$ trough the same network. Each such intermediate prediction $\hat{\mathcal{I}}^{k}$ results in a ℓ_2 loss term L_k, which is minimized during training. Further, we introduce novel temporal skip-connections to allow the network propagating information between different iterations potentially guiding subsequent iterations. Inspired by the work of Ronneberger *et al.* [RFB15] and the recent success of residual connections [He+16a] we use an encoder-decoder architecture in each deblur block, see Figure 6.14. Hereby, the network only consists of convolution and transpose-convolution layers with Batchnorm [IS15]. We applied the ReLU activation to the input of the convolution layers [He+16a].

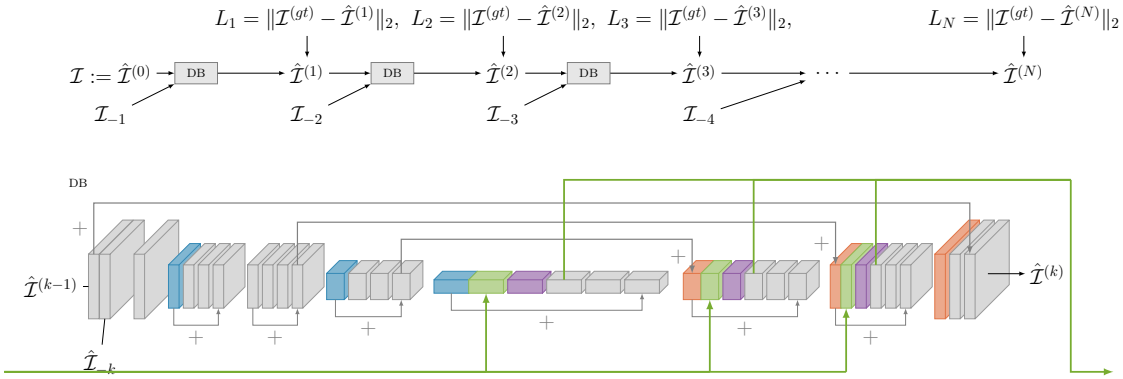


Figure 6.14: Given the current deblurred version of $\mathcal{I}$ each deblur block DB produces a sharper version of $\mathcal{I}$ using information contributed by another observation $\mathcal{I}_{-k}$. The deblur block follows the design of an encoder-decoder network with several residual blocks with skip-connections. To share learned features between various observations, we propagate some previous features into the current DB (green).

The first trainable convolution layer expands the 6-channel input (two 128×128px RGB images during training) into 64 channels. In the encoder part, each residual block consists of a down-sampling convolution layer (▪) followed by three convolution layers (▪). The down-sampling layer halves the spatial dimension with stride 2 and doubles the effective number of channels $[H, W, C] \to [H/2, W/2, C \cdot 2]$. During the decoding step, the transposed-convolution layer (▪) inverts the effect of the down-sampling $[H, W, C] \to [2 \cdot H, 2 \cdot W, C/2]$. We use a filter size of 3×3 / 4×4 for all convolution/transposed-convolution layers to avoid[2] checkerboard artifacts [ODO16]. In the beginning, an additional residual block without down-sampling accounts for resolving larger blur by providing a larger receptive field.

[2]Please refer to Section A for more details about such artifacts.

To speed up the training process, we add skip-connections between the encoding and decoding part. With this, we add the extracted features from the encoder to the related decoder part. This enables the network to learn a residual between the blurry input and the sharp ground-truth rather than ultimately generating a sharp image from scratch. Hence, the network is fully-convolutional and therefore allows for arbitrary input sizes. For more details please refer to Table 6.1.

Table 6.1: Encoder-Decoder Network Specification: Outputs of layers marked with * are concatenated with features from previous deblur blocks except in the first step. This doubles the channel size of the output. The blending layers "$B_{\cdot,\cdot}$" are only used after the first deblur step.

Layer	Filter-size	Stride	Output Shape
$A_{0,1}$	$3 \times 3 \times 64$	2	$H/1 \times H/1 \times 64$
$C_{1,1}$	$3 \times 3 \times 64$	2	$H/2 \times H/2 \times 64$
$C_{1,2}-C_{1,4}$	$3 \times 3 \times 64$	1	$H/2 \times H/2 \times 64$
$C_{2,1}-C_{2,4}$	$3 \times 3 \times 64$	1	$H/2 \times H/2 \times 64$
$C_{3,1}$	$3 \times 3 \times 128$	2	$H/4 \times H/4 \times 128$
$C_{3,1}-C_{3,4}$	$3 \times 3 \times 128$	1	$H/4 \times H/4 \times 128$
$C_{4,1}$	$3 \times 3 \times 256$	2	$H/8 \times H/8 \times 256$*
$B_{4,2}$	$1 \times 1 \times 256$	1	$H/8 \times H/8 \times 256$
$C_{4,3}-C_{4,5}$	$3 \times 3 \times 256$	1	$H/8 \times H/8 \times 256$

Encoder Architecture

Layer	Filter-size	Stride	Output Shape
$C_{5,1}$	$4 \times 4 \times 128$	1/2	$H/4 \times H/4 \times 128$*
$B_{5,2}$	$1 \times 1 \times 128$	1	$H/4 \times H/4 \times 128$
$C_{5,3}-C_{5,5}$	$3 \times 3 \times 128$	1	$H/4 \times H/4 \times 128$
$C_{6,1}$	$4 \times 4 \times 64$	1/2	$H/2 \times H/2 \times 64$*
$B_{6,2}$	$1 \times 1 \times 64$	1	$H/2 \times H/2 \times 64$
$C_{6,3}-C_{6,5}$	$3 \times 3 \times 64$	1	$H/2 \times H/2 \times 64$
$C_{7,1}$	$4 \times 4 \times 64$	1/2	$H/1 \times H/1 \times 64$
$C_{7,2}$	$4 \times 4 \times 6$	1	$H/1 \times H/1 \times 6$
$\mathcal{I}^{(k)}$	$3 \times 3 \times 3$	1	$H/1 \times H/1 \times 3$

Decoder Architecture

Skip connections as temporal links. We also propose to propagate latent features between subsequent deblur blocks over time. For this, we concatenate specific layer activations from a previous iteration with some from the current deblur block. These skip connections are illustrated as green lines in Figure 6.14. Further, to reduce the channel dimension to match the required input shape for the next layer, we use a 1×1 convolution layer, denoted as blending layer $B_{\cdot,\cdot}$. This way the network can learn a weighted sum by blending between the current features and propagated features from the previous iteration. This effectively halves the channel dimension. One advantage of such a construction is that we can disable these skip connections in the first deblur block and only apply these in subsequent iterations. Further, they can be applied to a pre-trained model without temporal skip connections.

Training details. Aligning inputs using homography matrices or estimated optical flow information can be error-prone and slows down the reconstruction preventing time-critical applications. Therefore, we trained the network directly on a sequence of unaligned frames featuring large camera shakes. To further challenge the network we add artificial camera shake to each blurry frame from synthetic PSF kernels on-the-fly. These PSF kernels of sizes $7 \times 7, 11 \times 11, 15 \times 15$ are generated by a Gaussian process (see Section 6.2.1) simulating camera shake. To account for the effect of vanishing gradients, we force the output $\mathcal{I}^{(k)}$ of each deblur block to match the sharp ground-truth $\mathcal{I}^{(gt)}$ in the corresponding loss term L_k (see Figure 6.14).

We use ADAM [KB14a] heuristic for minimizing the total loss $L = \sum_{k=1}^{4} L_k$ for sequences of 5 inputs with the optimizer's default parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and an initial learning rate $\eta = 5 \cdot 10^{-3}$.

6.4 Experiments

We evaluate the performance of our proposed method in several experiments on challenging real-world examples. In addition, a comprehensive comparison to recent methods is given using the implementation provided by the respective authors. Importantly, being fully convolutional the Recurrent Deblur Network (RDN) does not require patch-wise processing allowing to aggregate information between larger spatial distances and significantly speed up the reconstruction process. During inference we pass a pair of frames with resolution 1280×720px respectively 1920×1080px into a deblur block iteratively. Each iteration takes approximately 0.57 seconds on an NVIDIA Titan X. For any larger frame sizes, we tile the input frames. The network was trained exclusively on our synthetically blurred dataset featuring both motion blur and camera shake. All provided results in the section are based on benchmark sets from previous methods. Our RDN generalizes to different kinds of unseen videos and recording devices. Please note, we include the full-resolution images and frames from videos in the appendix (see Chapter C).

6.4.1 Burst Deblurring

In burst deblurring, the task is to restore a sharp frame from an entire sequence of aligned images. The sequence is usually taken by a single camera and only suffers from stationary blur caused by ego-motion. In our data-driven approach, we process each observation which finally produces significantly better results than previously proposed methods. Notably, ours is the first, which can restore the lettering below the license plate in Figure 6.15.

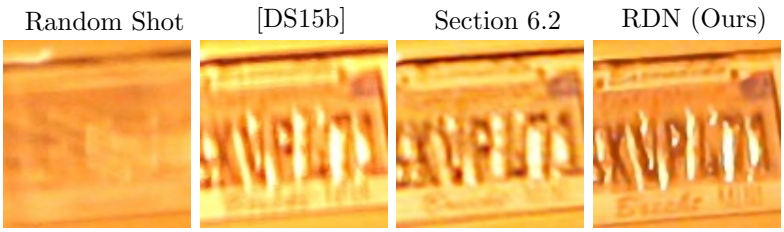


Figure 6.15: In contrast to previous state-of-the-art methods, our recurrent approach is able to even recover the subtle writing on the bottom of this number plate. It further reflects the original color tones from the random blurry shot. (Input image from [DS15a].)

Further, images featuring spatially varying blur are quite common in real-world scenarios due to imperfect lenses or turbulences. Figure 6.16 and 6.17 show a comparison to the Efficient Filter Flow framework (EFF) [Hir+10] which is dedicated this kind of blur. Notably, the results demonstrate two interesting findings of our approach: On one hand, our approach successfully generalizes to handle spatially varying blur, which was never explicitly provided in the training dataset and therefore the network was not trained on. On the other hand, although the network has been trained exclusively on training sequences of length 5, it can handle longer sequences and further improves the prediction due to its recurrent structure. However, too many input frames might introduce over-sharpening. And for longer sequences local contrast might saturate, potentially resulting also in a small color shift, which could be corrected using non-parametric histogram-based color matching.

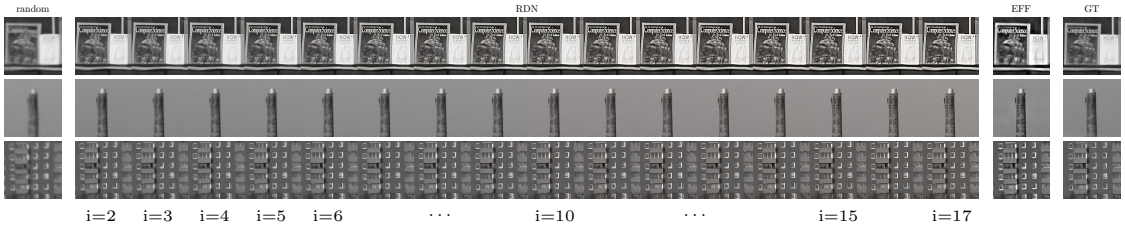


Figure 6.16: Recovery from image bursts with spatially varying blur. Reconstructions from using $i = 2, 3, \ldots$ blurry input frames are shown. A random shot from the inputs is given on the left and the ground-truth on the right next to the EFF result [Hir+10]. (Inputs images from [Hir+10].)

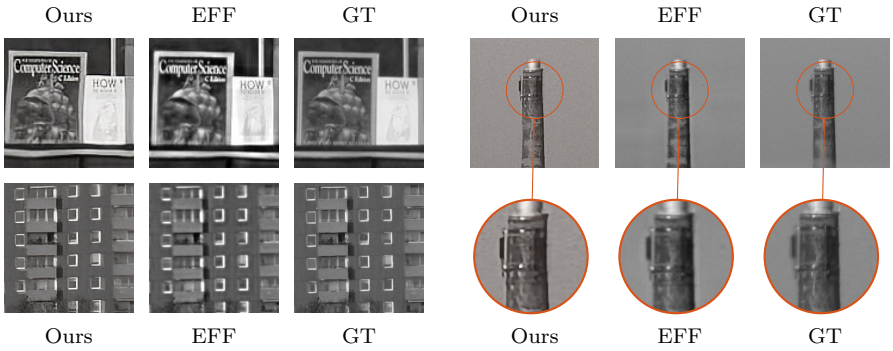


Figure 6.17: Zoomed-in version of final results from EFF [Hir+10] and our RDN compared to ground-truth (GT).

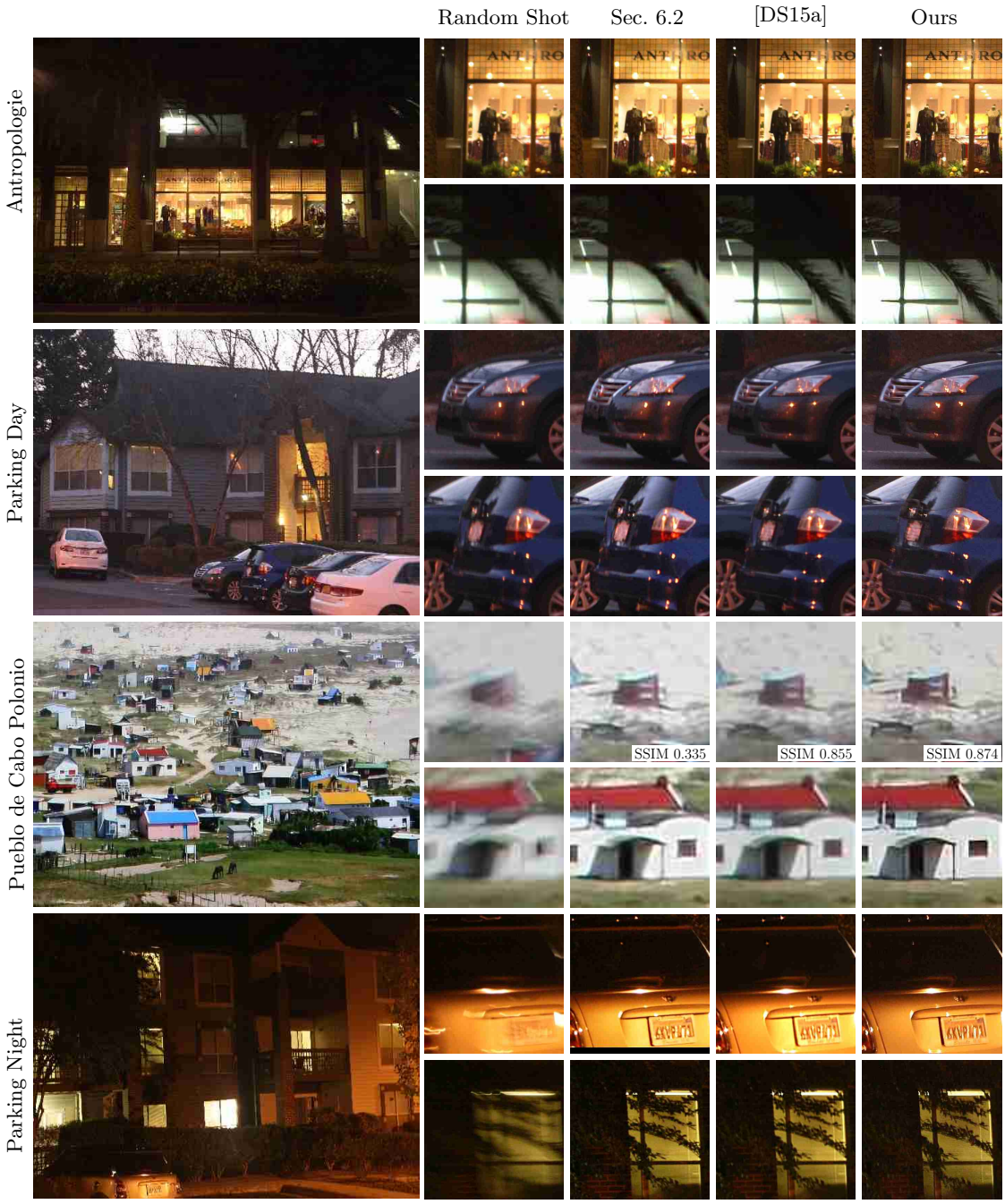


Figure 6.18: Comparison to state-of-the-art multi-frame blind deconvolution algorithms FourierNet ([Wie+16c], Section 6.2), FBA [DS15b] and ours (RDN) on real-world data for static scenes of low-light environments. RDN recovers significantly more detail. (Inputs images from [DS15a].) Best viewed in the electronic version by zooming in.

6.4.2 Video Deblurring

In contrast to the previous burst-deblurring task, videos are usually degraded by additional blur caused by object motion. Moreover, any deblurring approach has to solve the underlying frame alignment problem. Such an alignment step can be done offline, *e.g.* using a standard registration procedure, such as homography warping or by estimating optical flow fields to warp the frames to the reference frame. While this preprocessing delivers an easier task to the network, it might introduce artifacts which the network later has to account for. The approach of Su *et al.* [Su+17] (DBN) extensively use preprocessing (homography warping DBN(homog), optical flow alignment DBN(OF)) for alignment and directly train their networks to solve both tasks: deblurring and removing artifacts.

Our approach does not require any preprocessing. Hence, it is faster while producing comparable or better results. Figure 6.19 shows a comparison between DBN [Su+17] and our network directly applied to the input. Significant improvement in sharpness by our method can be observed on the trousers, the hair of the woman or the hand of the baby, to highlight a few. Artifacts due to the alignment procedure in the approach by Su *et al.* [Su+17] are visible on the lit wall in the Starbucks scene, for the cyclist in the second last row and in the piano scene, where the white keys are distorted (Figure 6.19, left second row). While ours is competitive when removing small motion, their optical flow based methods produce slightly sharper results when the camera motion is severe as seen on the road markings in the in the "bicycle" scene. Due to the limited capacity of the trained networks neither their nor our approach is fully capable of recovering the strong motion blur of swift motion.

Time-structure. Our network architecture consists of an "anti-causal" structure deblurring one frame by considering the original previous frames in a sequence-to-one mapping $\hat{\mathcal{I}} = DB(DB(\mathcal{I}, \mathcal{I}_{-1}), \ldots)$. We experimented with several sequence-to-sequence mapping approaches producing a sharp frame in an online way $\hat{\mathcal{I}}_t = DB(\mathcal{I}_t, \hat{\mathcal{I}}_{t-1})$. We noticed no learning benefit which might be caused by the limited capability of propagating temporal information.

Using multi-scale input. While our network has been trained on sequences of constant spatial resolution only, we experimented with feeding multi-scale input to recover strong object motion. In particular, we deblurred the entire input sequence at different levels $n = 1, 2, 3$ with $1/2^{n-1}$ resolution and then up-scaled the predicted result to obtain an additional new input frame for the sequence at the higher scale. While it partly helped to deal with larger motion blur which is not covered in the training data, the upsampling can produce artifacts which the network was not trained for. Figure 6.20 shows such results. Although the bike became significant

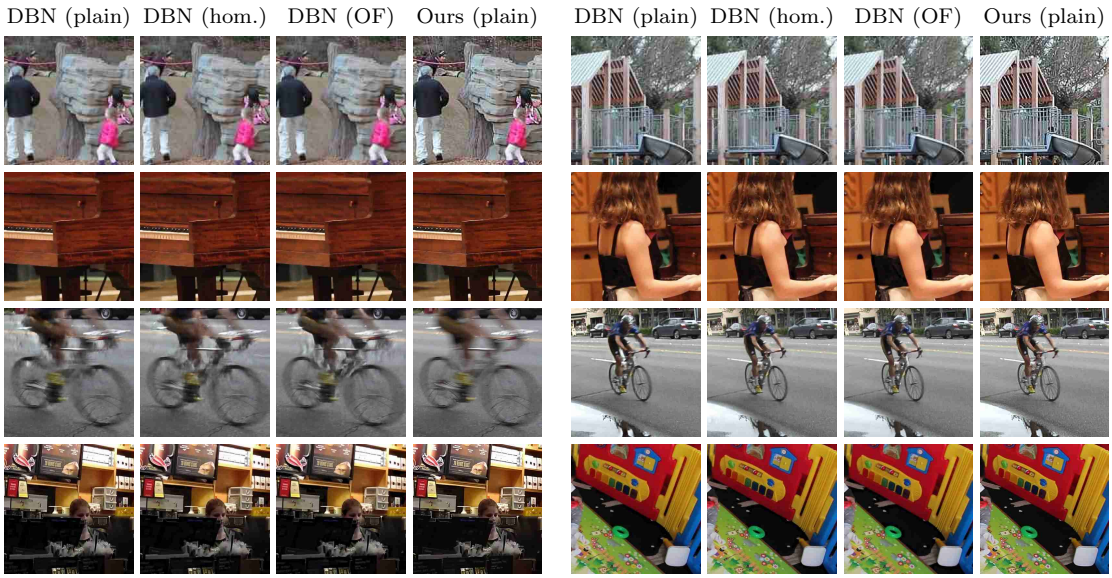


Figure 6.19: Comparison between (DBN) [Su+17] with different pre-processing strategies and our raw network prediction. (Inputs images from [Su+17].) Best viewed in the electronic version by zooming in.

sharper, the static parts of the scene rendered a "comic style" appearance. Directly training such a multi-scale network seems to be an interesting research direction.

Figure 6.20: Multi-scale input for large motion blur. We show the deblurred results with tradition single-scale input (1 level) or extending the input sequence with upscaled version of the deblurred results at half respectively quarter resolution.

Identifying valuable temporal information. One novel feature of our designed network architecture are the temporal skip connection (Figure 6.14 in green) acting as information links between subsequent deblur blocks. As we do not add constraints to these links, we essentially allow the network to propagate whatever feature information seems to be beneficial for the next deblur block. To illustrate this temporal information, we visualized the respective layer activation in Figure 6.21. The illustration suggests that the network uses this opportunity to propagate image locations which might profit from further deblurring (yellowish

parts). This is reasonable, as estimating the optical flow (bottom row) from blurry observations is as challenging as the deblurring task itself and flow information can only indirectly help to solve in motion deblurring.

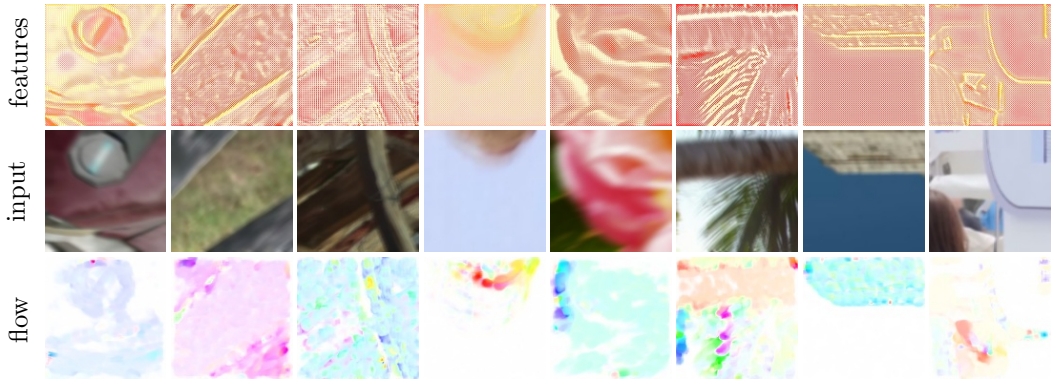


Figure 6.21: Visualization of features propagated along the temporal skip connections (top-row) when encoded in HUE colorspace, random input image (mid row) and estimated optical flow used exclusively to synthesize the motion blur (bottom row). Apparently, the network learned to mark regions in "features" which might benefit from deblurring.

6.5 Possible Future Work

Deblurring videos introduces the challenge of spreading information across the temporal domain over potentially hundreds of frames. However, training such long sequences is not feasible considering the computational costs. The introduced iterative approach (Section 6.3.3) generalizes from a 5-tuple to inputs lengths never observed during training (Figures 6.16, 6.17) before color shift artifacts occurs.

Retrospectively, the entire network can be seen as a more extensive version of an LSTM [PHC16] or GRU cell [Cho+14] without explicit gates. A desirable property of a neural network architecture handling video inputs for tasks like deblurring or super-resolution would be a baked-in mechanism (gates) enabling to weight new input observation regarding their estimated impact on the prediction. Such a network would be versed to filter out mediocre observations and could stop consuming new input-observations if the expected output quality would decrease. A possible lead would be to start learning to rank images according to their estimated blurriness.

A significant issue in dynamic scenes is handling video-frames of moving objects. While the previous section demonstrated a neural network is capable of detecting and sharpening such objects, we hypothesize that learning to estimate the optical flow and warping the features will ease the task of broadcasting information over

longer temporal distances. Along with temporal propagation, a deep-learning approach is required to take information from different spatial locations into account. Aliasing and blur effects could be more effectively solved in a repeating texture if such pattern is detected. Current architectures prefer smaller filter-kernel sizes 3×3 for their smaller computational costs but therefore lack the size of the receptive fields. Instead of using separable filters [Xu+14] or multi-scale approaches like ours, classical algorithms exploit patch recurrence [MI14]. Enabling convolutional neural networks to match similar features across different scales or within multiple inputs, *e.g.* using PatchMatch [Bar+09] algorithm, would overcome the issue of the receptive field with a limited spatial size and would allow connecting features across the entire image.

Another interesting idea, which has been proven to be successful in intrinsic images decomposition is the introduction of an auxiliary task like edge prediction [Fan+17b]. Thereby, a network-part is pre-trained to predict a sparse representation of the final reconstruction. Given such an intermediate solution of the "guidance network" a second network is then trained to refine the sparse representation and predict the reconstruction. The motivation of such an approach is the empirical evidence [Dos+15] that several refinement networks and redundant inputs [Ilg+17] eases the training.

As the camera shake is unknown, the spatial offsets between consecutive frame are unkown as well. However, in multi-view stereo 3D reconstructions the baseline between two cameras might be known beforehand. While our trained network currently does not exploit such information, it already delivers useful reconstruction of image-triplets compare to our (yet unpublished) multi-scale encoder-decoder network trained on single image inputs for two months, see Figure 6.22. This figure additionally confirms the need for multi-frame methods.

Explicitly exploiting the (known) baseline information of stereo-cameras enrich the input information and therefore is likely to improve the quality further. Such a camera setup is rather typical, considering the latest development in the smartphones industry. These consumer devices already carry two lenses of different aperture and a small baseline, *e.g.* for faking depth of field.

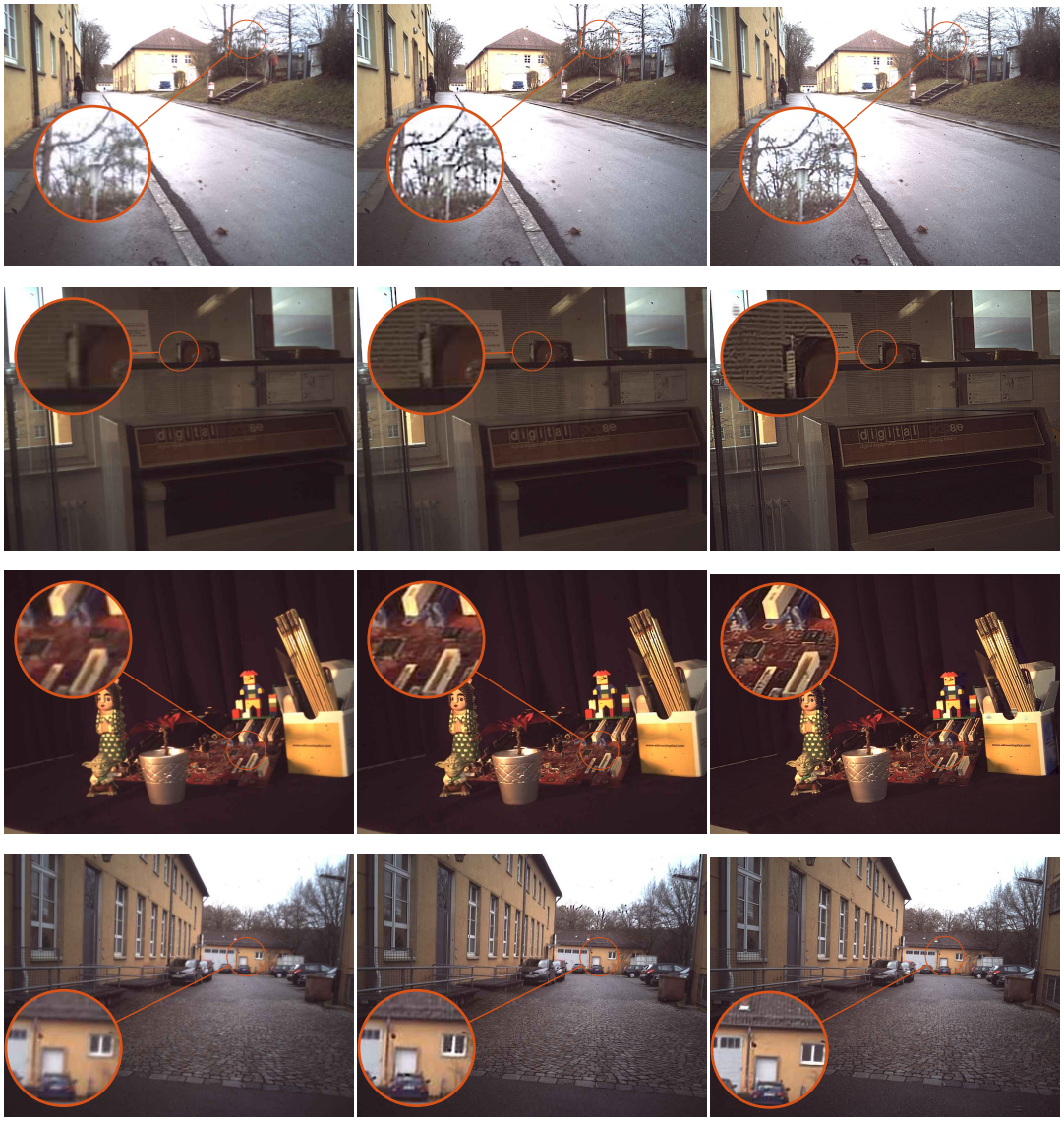


Figure 6.22: Comparison between one input (left), a network trained over two months (middle) on single-image deblurring and our multi-frame approach (right). The proposed multi-frame method shows clear advantages in resolving and deblurring filigree details as visible in the branching of the trees, the characters in the info text, electronic circuit details and roof tiles. (Inputs images were captured by Benjamin Resch.)

Rule #20:
The training time does not measure the convergence rate — but your patience.

Chapter 7

Conclusion

The application of deep learning methods is one essential technique to solve fundamental problems in computer vision. This thesis has demonstrated several ways beyond the processing of single image inputs and showed severe improvements in the performance by incorporating multiple observations in novel multi-frame methods.

Chapter 2 demonstrates that a vital component of computer vision pipelines, nearest neighbor search, can be efficiently solved on modern hardware like GPUs. Improvements to the algorithm itself by cutting down the number of exact vector comparisons and parallelizing approximated sequential searches during ranking had tremendous effects on performance. Although our algorithm is tailored to GPU hardware, we have shown that even our CPU implementation achieves state-of-the-art results, while our GPU version significantly surpasses the performance of earlier work on standard benchmark datasets.

In Chapter 4, it turned out that the temporal information carried over several successive frames is sufficient to learn a global scene descriptor without any external supervision from first principle: "Consecutive observations are similar". Indeed, this learned descriptor is even resistant to alterations in appearance and has been trained from a rather large but affordable dataset. This makes it quite evident how the training of modern artificial neural networks can be summed up by the question of how to draw training samples efficiently. Further, we introduced a heuristic to reject wrong predictions to automatically compile a more complex and reliable training dataset on-the-fly, from which a improved model is iteratively derived.

Chapter 5 was aimed at the blind signal separation of reflection and transmission in uncontrolled environments. This revealed another strength of multi-frame methods based on incorporating physical models as a novel initial layer in combination with a learned image prior. We illustrated a way of synthesizing physically plausible training data, which incorporates imperfections of the capturing process itself, like alignment issues from motion. We empirically proved all these subtle ingredients of our data generation pipeline are necessary to improve the results on real-world examples compared to previous work. While reflections usually hinder accurate 3D

acquisition, such a separation could indeed help to even recover structures which are not directly observable.

In Chapter 6 we discussed the problem of blind image deconvolution using multiple observations. We started by showing that a deep neural network can benefit from the integration of modified classical methods. Therefore, we adapted a lucky-imaging method such that it is suitable as a neural network layer, which can be jointly trained in an end-to-end fashion while leading to a substantial improvement over the combination of previous state-of-the-art methods. In a second approach, we proposed a novel network architecture that can flexibly handle different lengths of input sequences, eliminating the limitation of handling only a certain number of observations. We saw empirical evidence that our present recurrent deblurring blocks can readily generalize to much longer sequences than observed during training. Although, we trained our deep-learning approach exclusively on spatially-invariant blur the achieved results on spatially-variant blur is at least on-par with state-of-the-art methods.

All these introduced approaches do not involve any tricks like excessively tuning hyperparameters or resorting to generative adversarial models (GAN) that would hallucinate any missing information and break evaluations protocols based on standard metrics like PSNR or RMSE. The main part in each solution is based on an effective sampling of training data: "*When to use a data point?*" (Chapter 4) or "*How to realistically synthesize training data?*" with all imperfections given a limited amount of available ground-truth data (Chapters 5 and 6) or even in the absence of it.

The multi-frame methods proposed here have been proven successful in advancing the state-of-the-art in performance and flexibility. Nevertheless, open problems remain. First and foremost, recent hardware developments allow to manifest observations in digital form in very high resolution and high framerate. There already exists devices which can capture data with 60fps in 16k resolution. Today's deep learning methods can barely handle such data, which is a severe limitation. One reason is that, while basic convolution layers are spatially-variant, they lack concerning resolution not being scale-invariant.

A significant development, besides the possibility that deep neural networks can treat many observations at low latency, must be the capability of processing with higher resolution. Prediction of deep neural networks at VGA resolution must no longer be appropriate.

Rule #1:
It is wise to be deeply suspicious of any deep learning method – even your own.

Appendix A

Artifacts in Outputs of Deep Neural Networks

When training CNNs on image-related tasks like image deblurring, the predicted image from a deep neural network can suffer from unwanted effects like checkerboard artifacts, ringing artifacts or even color desaturation issues. These distortions of the output delivering mediocre results can be corrected, reduced or even suppressed by the following design choices in the neural network architecture.

A.1 Checkerboard Artifacts

Output images from neural networks can exhibit the checkerboard pattern even in supposedly homogeneous image regions. In Figure A.1 an encoder-decoder network has been trained to reconstruction the input images. The encoder consists of convolution layers with kernel-size 3 and stride 2 for sub-sampling purposes. Mirroring the structure of the encoder network in the decoder network by using transposed convolution layers with kernel-size 3 and stride 2 can result in reconstructions which contain a regular high-frequency pattern as shown in Figure A.1. When using kernel-size 4 instead of 3 in the transposed convolution layers the output quality unequivocally is improved.

This effect can be intuitive explained in the case of a transposed convolution layer containing only filter-weights with value 1. Applying such a transposed convolution layer to an input image with constant pixel values, *e.g.* all set to 1, illustrates the effect. Figure A.2 contains the output of chaining two transposed convolution layers with kernel-size 3 (left) and kernel-size 4 (right).

As this effect is "baked" into the neural network design, choosing kernel-size 3 in transposed convolution layers will have an impact on the neural network performance and training efficiency as the neural network is constantly optimized to avoid these inherent patterns. It is common practice to consider a nearest-neighbor up-sampling. However, such an un-parametrized layer cannot be trained reducing the flexibility of the neural network. Even worse such an up-sampling strategy will up-sample any features without correcting them such that following layers will

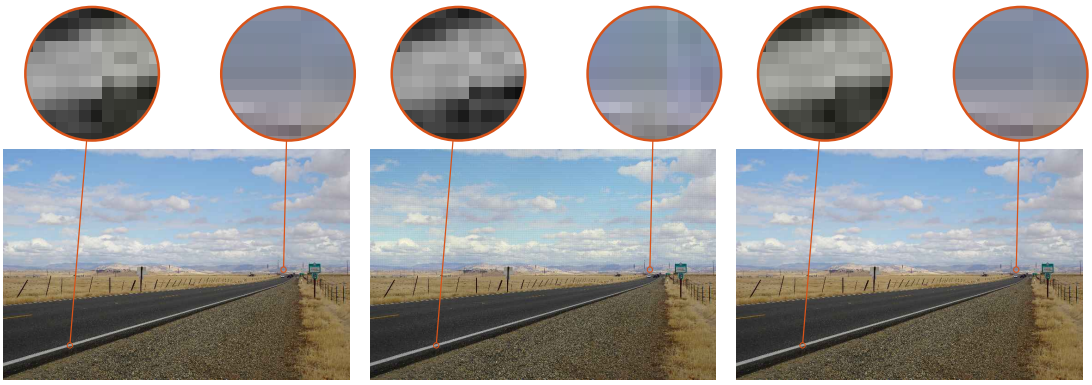


Figure A.1: A encoder-decoder neural network has been trained to reconstruct the input image (without Unet skip-connections). Left: An example input image shown to the neural network. Middle: Output of the neural network, where the decoder network part mirrors the encoder network and kernel size 3 for the reconstruction process. Right: Output of the neural network, with the same decoder network structure but using kernel-size 4.

require a larger filter size to ensure potential miss-placed feature can be detected within the receptive field. This is the motivation to use transposed convolution layers instead of nearest-neighbor up-sampling throughout all data-driven approaches in this thesis.

A.2 Color Desaturation Artifacts

Another quite common issue when training deep-neural network is missing saturation of the colors in the predicted image. Such an example has been shown in Section 6.2 for image deblurring of static scenes (see Figure 6.6) and in Chapter 5 for reflection removal. There exists several different strategies to correct for this issue. In Chapter 5 a non-parametric histogram-matching of the colors in the output image is used to match those in the input image as an independent post-processing step during inference. And Section 6.2.3 describes our approach of replacing the colors from the predicted image by the colors of the input image in CIE-*Lab* color space. The RDN in Section 6.3.3 has been as well been a victim of this effect during early stages of the training. Empirical observations show the evidence that these desaturation issues can be removed by longer training runs of the deep neural network. Preliminary tests for the RDN in Section 6.3.3 suggest that using ReLU or omitting the activation function in the last output layer greatly reduces the occur-

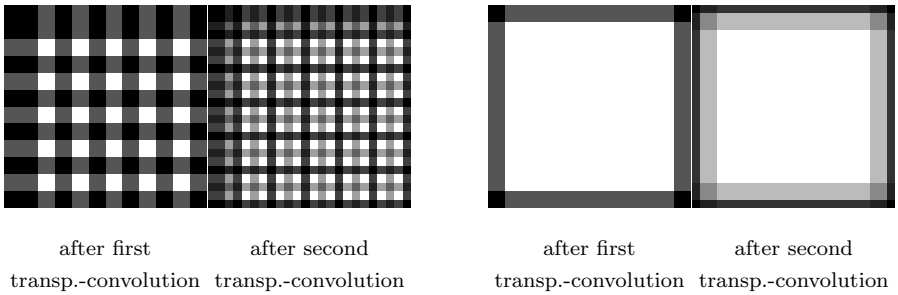


Figure A.2: Applying a two stacked transposed convolution layers with weights all set to 1 to an input image with all pixels set to 1. Left: The output of both layers with kernel-size set to 3. Right: The outputs of the stacked transposed convolution layers when kernel-size is set to 4. The contrast and size of these images has been increased for illustration purposes.

rence of color desaturation compared to other non-linearities like tanh or Sigmoid. This seems to be counterintuitive as tanh and the Sigmoid non-linearity function acts a safeguards to ensure the predicted values are indeed within a meaningful range for pixel intensities (*e.g.* typically $[0, 1]$ or $[-1, 1]$).

My conjecture for this artifact being more common when using a non-linearity as the final activation function is based on the observation that in early trainings iterations the mean RGB value of the predicted image is wrong, which is one form of desaturated colors. And the neural network cannot correct this shift of the mean output value in this case.

Any clipping or clamping operation by the activation function in the final layer will unintentionally hide high costs for large values by the nature on how these non-linear mappings work. Figure A.3 shows two common activation functions used in neural networks as a final activation function. Let $\varepsilon > 0$. Predicting the value $x + \varepsilon$ instead of the correct value x would cause only small gradients in the regions marked as blue, while the training signal is significant higher in the red area. The gradient which scales the back-propagation signal during training (see Section 3) is much higher in the red-marked region. In other words, squeezing the valid range of output values into a pre-defined interval (*e.g.* in the range $[-1, 1]$) using a non-linear mapping reduces costs a neural network has to pay for difference in extreme cases and therefore causes a weaker training signal. Hence, the neural network can afford to wrongly predict lower and higher intensities values compared to values in the mid of the valid range.

But these non-linearities are still commonly used. One reason for their popularity in GANs (*e.g.* see CycleGAN [Zhu+17] and related work) is their guarantee for pixels from the generator to be within the correct range eliminating the short-cut for the discriminator to judge from the value range itself. Otherwise, the discriminator

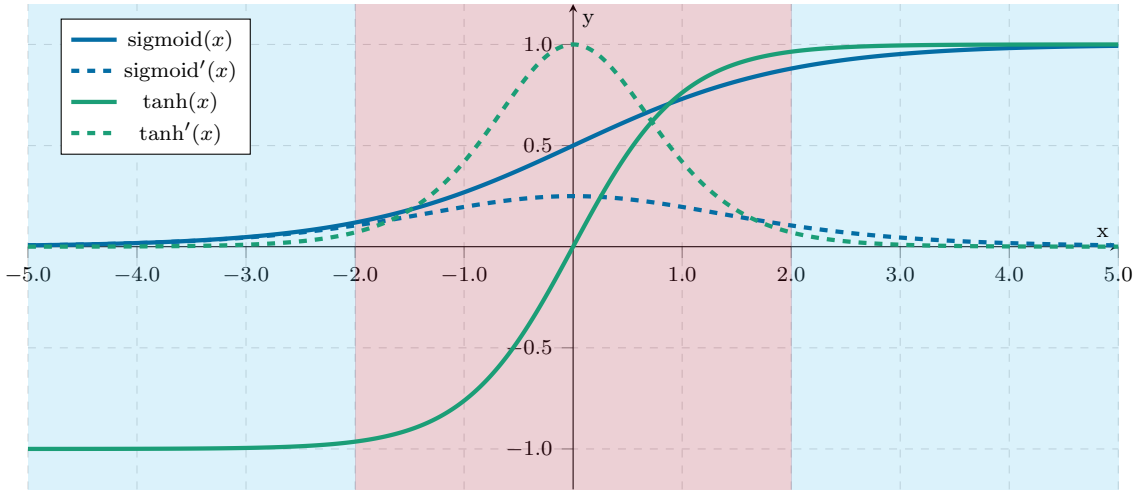


Figure A.3: Typical activation functions and their derivatives. Each derivative features a "bump" which causes a strong gradient in the red-marked region and weak gradients otherwise.

would not provide a meaningful back-propagation signal to the generator during training.

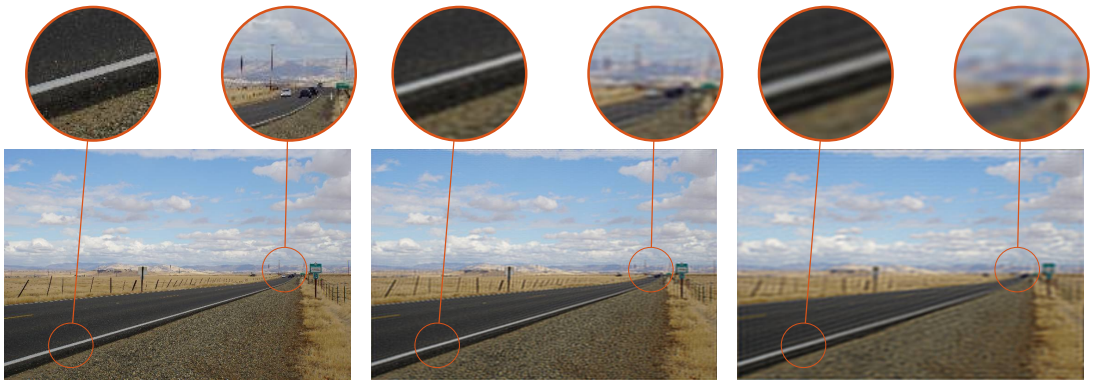
A.3 Ringing Artifacts

Another common artifact in predictions from deep neural networks – especially in image deblurring – are ringing artifacts. These effects are quite noticeable in early iterations during training. Strong edges in images might exhibit a ghosting. An illustration of these effects is given in Figure A.4. The road markings edges appear multiple times in the image and fine structures as shown in the background in Figure A.4 are likely to produce blob-like artifacts. One cause of this effect is high-frequency information corruption. In Figure A.4, we trained an encoder-decoder neural network on restoring the sharp image from a blurry observation. Since the expressiveness of the neural network was radically reduced, only low-frequency information can be reconstructed by the neural network. Depending on the number of layers this effect might occur with noticeable larger edge ghosting or smaller ghosting effects.

This effect behaves like a low-cut filter in Fourier space and can be synthetically reproduced: Let x be the input signal and m_r be the mask with $m_r(x, y) = 1$ if and only if $\sqrt{x^2 + y^2} \leq 1$ else 0. Then the effect can be simulated by

$$y = \mathcal{F}^{-1}\left(m_r \odot \mathcal{F}(x)\right), \tag{A.1}$$

where $\mathcal{F}$ denotes the discrete 2D Fourier-Transform. An illustration is given in Figure A.5.



Input | Small Ringing Artifacts | Larger Ringing Artifacts

Figure A.4: Two encoder-decoder neural networks (6 (middle) resp. 4 (right) stages in encoder and decoder) have been trained to deblur the input image (without Unet skip-connectins) with sharp ground-truth image (left). Both networks feature ringing artifacts.

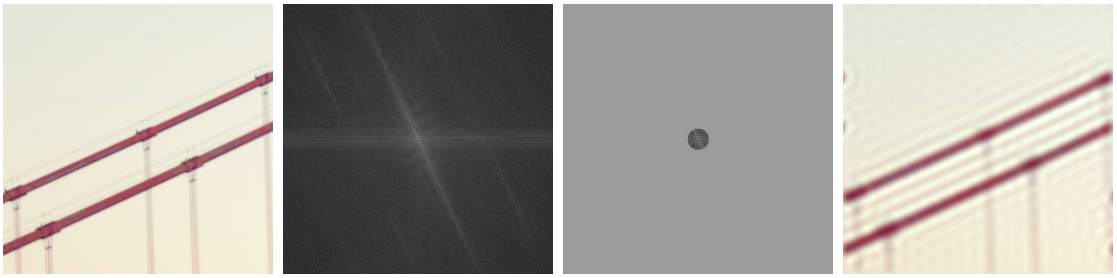


Figure A.5: For a given input image (left) cutting high-frequencies in the Fourier space (middle) and transforming the remaining information back leads to ringing artifacts (right).

The absence of high-frequency information is particularly true in early training iterations of the Burst-Deblurring Network in Section 6.2 as the neural network uses the Fourier transform of the input.

A possible direction in future work is to train a deep neural network with two loss functions at different positions in the neural network: One loss function to force reconstructing the low-frequency information potentially without ringing artifacts and a GAN-like approach to fill in the high-frequencies to improve perceptual quality. But training a deep neural network to "just" reconstruct low-frequency information without ringing artifacts is possible as the neural network usually directly operations in the spatial domain.

Rule #34:
You cannot advance the field by sticking to the usual layers. We simply need other operations in the neural network.

Appendix B

Appendix for Separating Reflection and Transmission Images

The following section presents more details on the described approach for separating reflection and transmission images (see Chapter 5). The used encoder-decoder neural network architecture is given as below.

```
lr = symbolic_functions.get_scalar_var('learning_rate', 5e-3, summary=True)
optimizer = tf.train.AdamOptimizer(lr)

# project onto canonical axes
I_s, I_p = img_otho_extraction(I1, I2, I3)
observations = tf.concat([I_s, I_p, I1, I2, I3], axis=3)
observations = observations * 2. - 1.

# ResNet block for down-sampling
def block_down(net, nf, name, stride=2):
    with tf.variable_scope(name):
        skip = Conv2D('conv1', net, nf, stride=stride, nl=tf.identity)
        net = INReLU('inrelu', skip)
        net = Conv2D('conv2', net, nf)
        net = Conv2D('conv3', net, nf)
        net = tf.concat([net, skip], axis=3)
        net = Conv2D('conv4', net, nf, kernel_shape=1)
        return net

# ResNet block for up-sampling
def block_up(net, uskip, nf, name, stride=2):
    with tf.variable_scope(name):
        skip = Deconv2D('conv1', net, nf, kernel_shape=4,
                        stride=stride, nl=tf.identity)
        net = INReLU('inrelu', skip)
        net = tf.concat([net, uskip], axis=3)
```

```
        net = Conv2D('conv2', net, nf)
        net = Conv2D('conv3', net, nf)
        net = tf.concat([net, skip], axis=3)
        net = Conv2D('conv4', net, nf, kernel_shape=1)
        return net

# entire network architecture (INReLU is InstanceNormalization with ReLU)
with argscope([Conv2D, Deconv2D], nl=INReLU, kernel_shape=3, stride=1):
    net = observations

    # prologue
    out0 = Conv2D('conv0', net, 15)

    # encoder
    out1 = block_down(out0, 32, 'block1')
    out1b = block_down(out1, 32, 'block1b', stride=1)
    out2 = block_down(out1b, 64, 'block2')
    out2b = block_down(out2, 64, 'block2b', stride=1)
    net = block_down(out2b, 128, 'block3')
    net = block_down(net, 128, 'block3b', stride=1)

    # decoder
    net = block_up(net, out2b, 64, 'block4')
    net = block_up(net, out2, 64, 'block4b', stride=1)
    net = block_up(net, out1b, 32, 'block5')
    net = block_up(net, out1, 32, 'block5b', stride=1)
    net = block_up(net, out0, 16, 'block6')

    # epilogue
    with tf.variable_scope('epilog'):
        net = Conv2D('deconv_1', net, 16)
        net = tf.concat([net, observations], axis=3)
        net = Conv2D('deconv_2', net, 16)
        net = Conv2D('deconv_0', net, 8, kernel_shape=3,
                     stride=1, nl=tf.identity)

    # gates
    mask_t = tf.expand_dims(tf.sigmoid(net[:, :, :, -1]), axis=-1)
    mask_r = tf.expand_dims(tf.sigmoid(net[:, :, :, -2]), axis=-1)

    # direct estimation
    pre_t = (tf.tanh(net[:, :, :, :3]) + 1.) / 2.
    pre_r = (tf.tanh(net[:, :, :, 3:6]) + 1.) / 2.
```

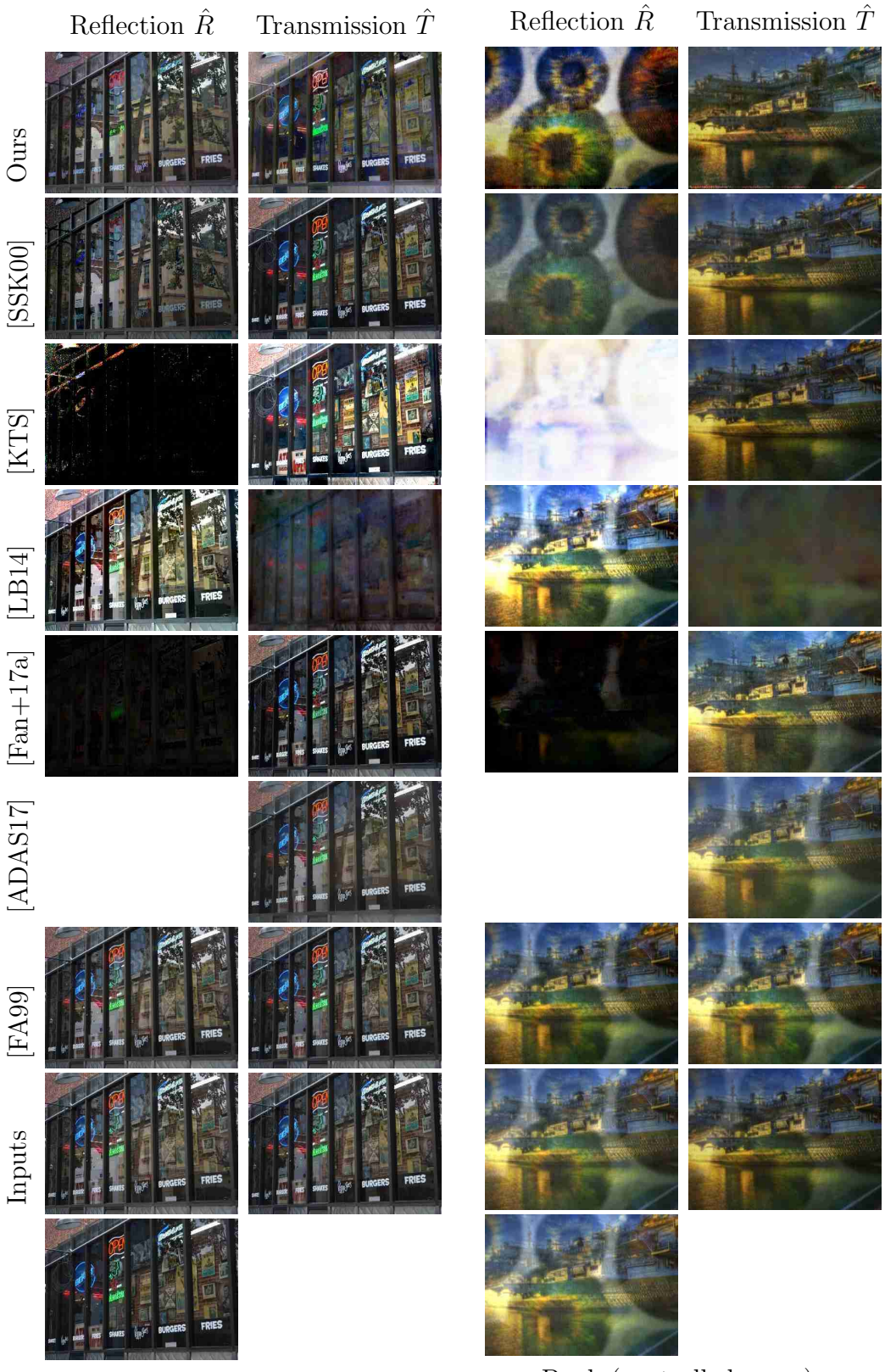
```
# blending
est_lr = mask_r * pre_r + (1 - mask_r) * I_s
est_lt = mask_t * pre_t + (1 - mask_t) * I_p
```

The remaining part of this section contains an extensive comparison of the algorithm presented in Chapter 5 against previous work [SSK00; KTS; LB14; Fan+17a; ADAS17; FA99] for separating reflection and transmission images on URD.


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Curved window
Diner

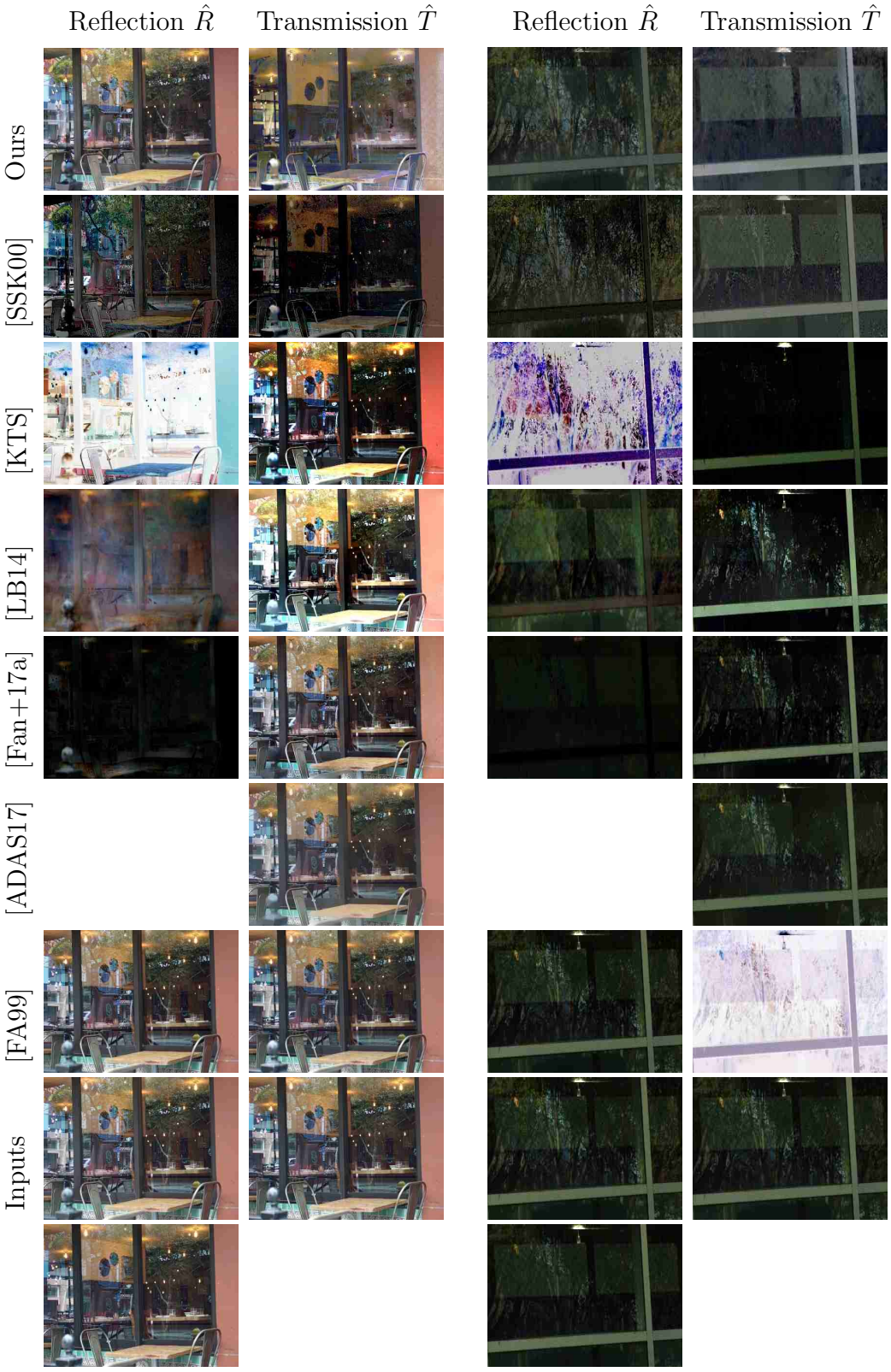

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Ice cream shop
Ticket counter

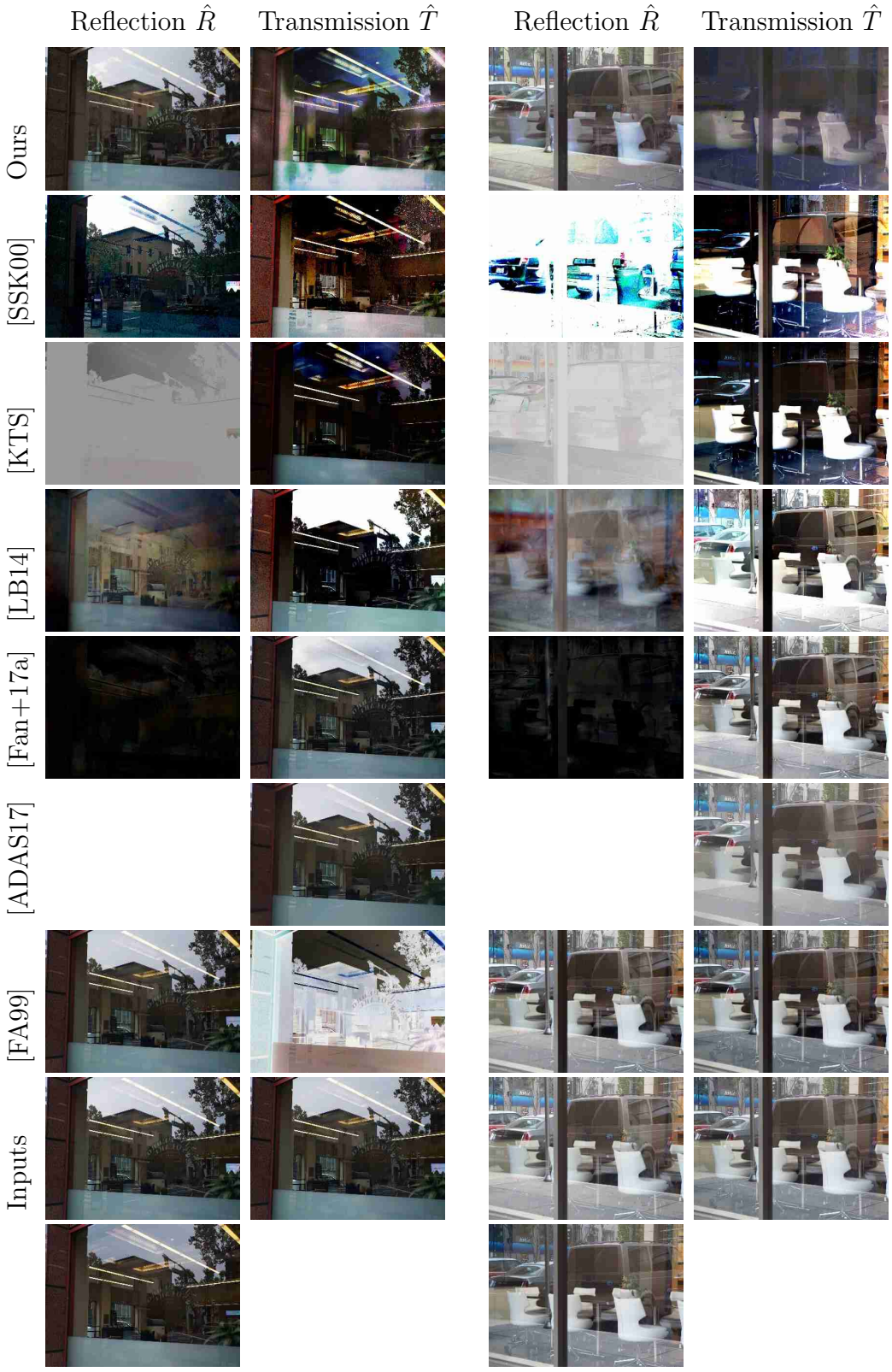

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Wall
Gym


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Bar
Book (controlled scene)

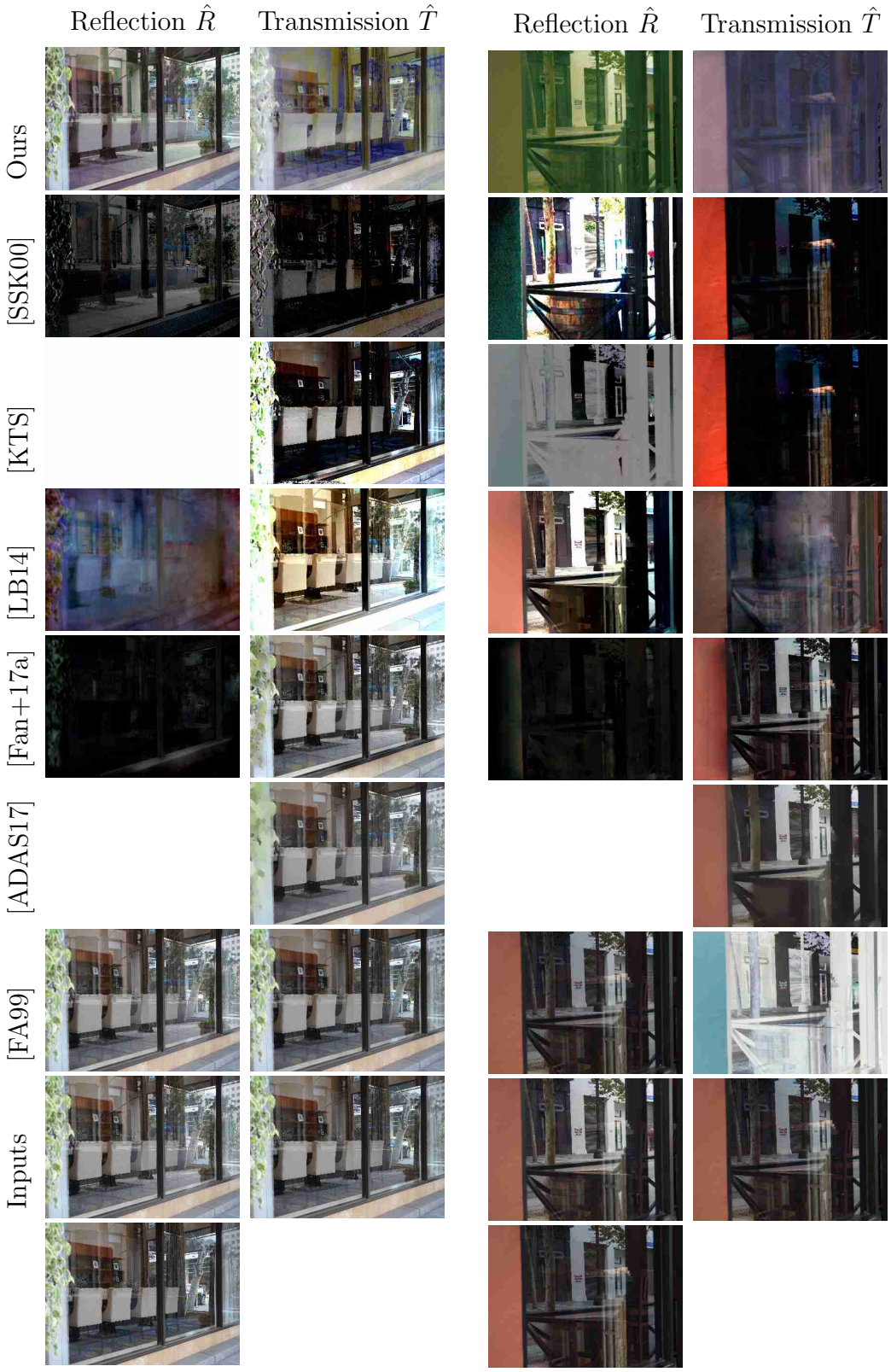

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Painting
Knoll

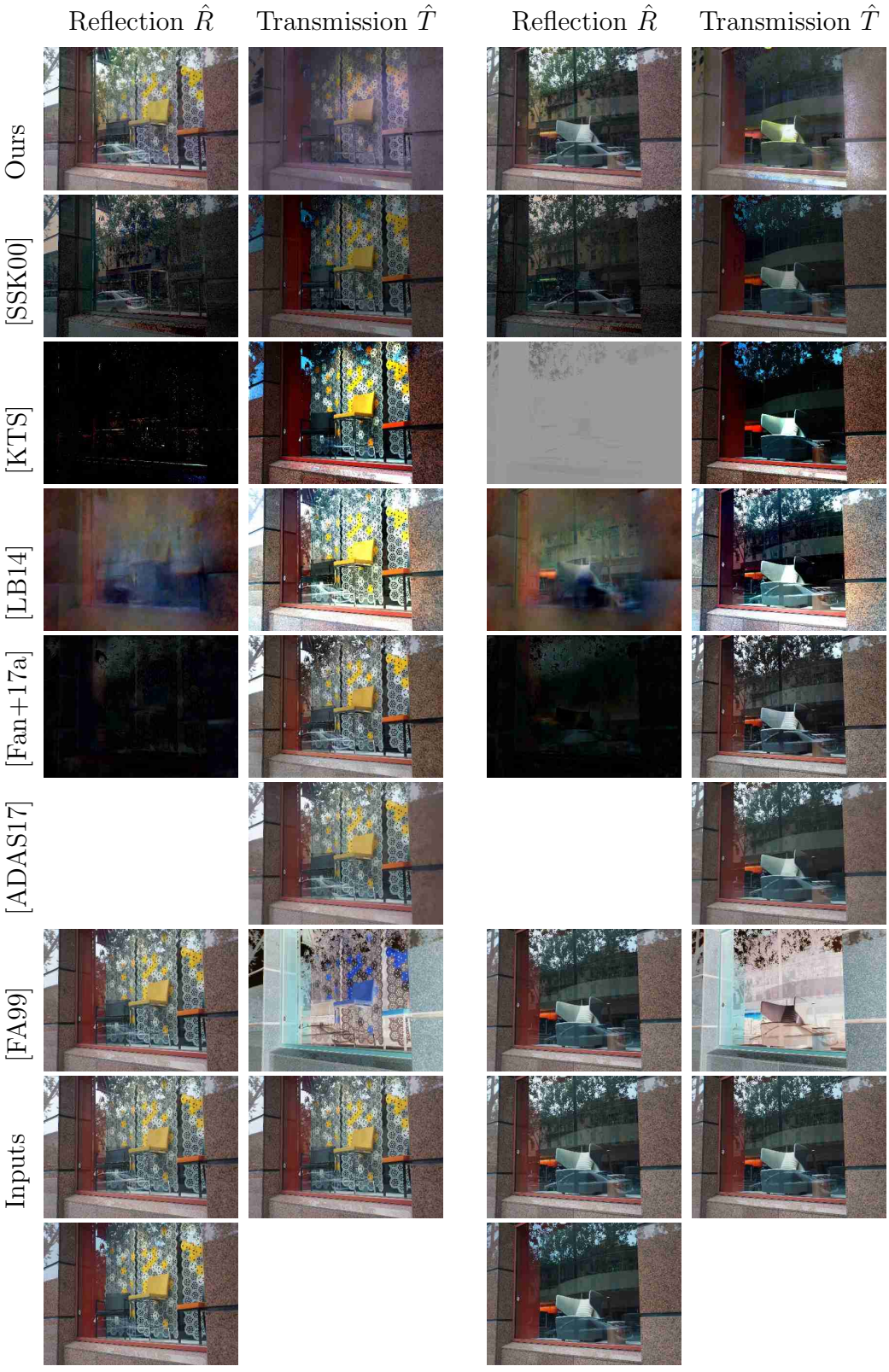

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Bookshelf
Sidestreet

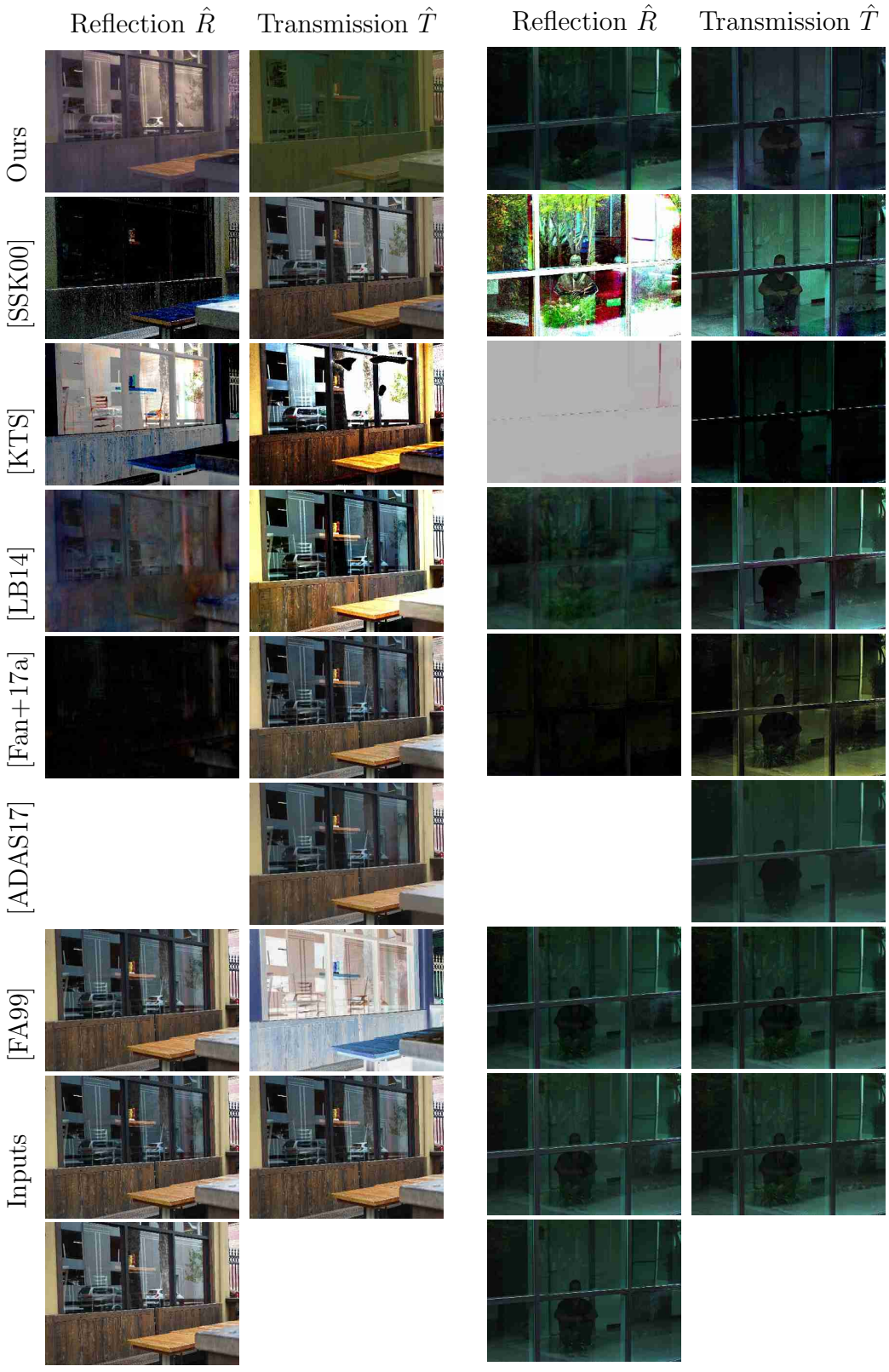

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Restaurant
Window

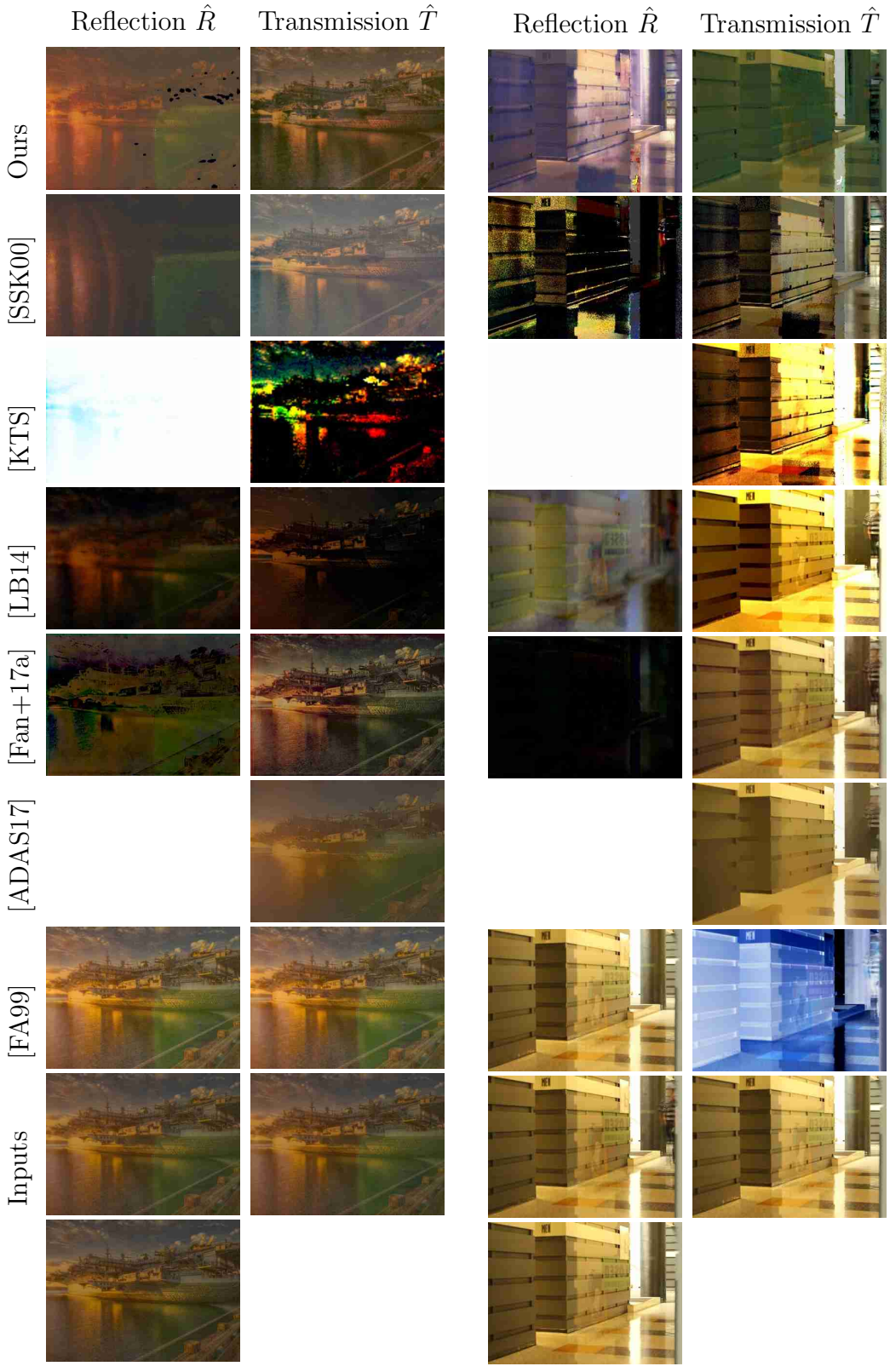

Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Market
Conference


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Entrance
Garage


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Library
Barrol


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Chair
Chair 2


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Diner 2
Ghosting-cues


Reflection $\hat{R}$
Transmission $\hat{T}$
Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs
Toys
Floor


Reflection $\hat{R}$
Transmission $\hat{T}$
Ours
[SSK00]
[KTS]
[LB14]
[Fan+17a]
[ADAS17]
[FA99]
Inputs

Car

Rule #46:
Proper training data is a necessary but not sufficient condition for a decent performance of a neural network.

Appendix C

Appendix for Multi-Frame Deblurring

The following section presents a comparison of the approach proposed in Chapter 6 against the work of Su *et al.* [Su+17] in higher resolution. Images were taken from Su *et al.* [Su+17] .

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

plain [Su+17]

with homography pre-processing [Su+17]

with optical flow pre-processing [Su+17]

Ours

The used deblur block as part of the used neural network architecture is given as below.

```
@auto_reuse_variable_scope
def deblur_block(self, observation, estimate,
                 skip_temporal_in=None,
                 name=None):
    """Apply one deblur step.
    Args:
        observation: new unseen observation
        estimate: latest estimate (the image which should be improved)
        skip_temporal_in (None, optional): list of skip_connections
        skip_unet_out(None, optional): list of skip connections
           between deblurring blocks within the network .
    """

    skip_temporal_out = []  # green
    skip_unet_out = []      # grey

    with tf.name_scope("deblur_block_%s" % name):
        # be aware use_local_stat=True gives warnings
        with argscope(BatchNorm, use_local_stat=True), \
                argscope([Conv2D, Deconv2D], nl=BatchNorm):
            inputs = tf.concat([observation, estimate], 3)

            block = ReluConv2D('d0', inputs, 32, stride=1, size=3)

            # H x W -> H/2 x W/2
            # ------------------------------------------------------------
            with tf.name_scope('block_0'):
                block = ReluConv2D('d1_0', block, 64, stride=2)
                block_start = block
                block = ReluConv2D('d1_1', block, 64)
                block = ReluConv2D('d1_2', block, 64)
                block = ReluConv2D('d1_3', block, 64, size=1)
                block = tf.add(block_start, block, name='block_skip_A')

            # H/2 x W/2 -> H/2 x W/2
            # ------------------------------------------------------------
            with tf.name_scope('block_1'):
                block = ReluConv2D('d2_0', block, 64)
                block_start = block
                block = ReluConv2D('d2_1', block, 64)
                block = ReluConv2D('d2_2', block, 64)
                block = ReluConv2D('d2_3', block, 64, size=1)
```

```
        block = tf.add(block_start, block, name='block_skip_B')
        skip_unet_out.append(block)

    # H/2 x W/2 -> H/4 x W/4
    # ----------------------------------------------------------
    with tf.name_scope('block_2'):
        block = ReluConv2D('d3_0', block, 128, stride=2)
        block_start = block
        block = ReluConv2D('d3_1', block, 128)
        block = ReluConv2D('d3_2', block, 128)
        block = ReluConv2D('d3_3', block, 128, size=1)
        block = tf.add(block_start, block, name='block_skip_C')
        skip_unet_out.append(block)

    # H/4 x W/4 -> H/8 x W/8
    # ----------------------------------------------------------
    with tf.name_scope('block_3'):
        block = ReluConv2D('d4_0', block, 256, stride=2)
        block_start = block
        block = Merge(skip_temporal_in, 0, block, 'd41_s')
        block = ReluConv2D('d4_1', block, 256)
        block = ReluConv2D('d4_2', block, 256)
        block = ReluConv2D('d4_3', block, 256, size=1)
        block = tf.add(block_start, block, name='block_skip_D')
        skip_temporal_out.append(block)

    # H/8 x W/8 -> H/4 x W/4
    # ----------------------------------------------------------
    with tf.name_scope('block_4'):
        block = ReluDeconv2D('u1_0', block, 128, stride=2, size=4)
        block = tf.add(block, skip_unet_out[1], name='skip01')
        block_start = block
        block = Merge(skip_temporal_in, 1, block, 'u1_s')
        block = ReluConv2D('u1_1', block, 128)
        block = ReluConv2D('u1_2', block, 128)
        block = ReluConv2D('u1_3', block, 128)
        block = tf.add(block, block_start, name='block_skip_E')
        skip_temporal_out.append(block)

    # H/4 x W/4 -> H/2 x W/2
    # ----------------------------------------------------------
    with tf.name_scope('block_5'):
        block = ReluDeconv2D('u2_0', block, 64, stride=2, size=4)
        block = tf.add(block, skip_unet_out[0], name='skip02')
```

```
        block_start = block
        block = Merge(skip_temporal_in, 2, block, 'u2_s')
        block = ReluConv2D('u2_1', block, 64)
        block = ReluConv2D('u2_2', block, 64)
        block = ReluConv2D('u2_3', block, 64)
        block = tf.add(block, block_start, name='block_skip_F')
        skip_temporal_out.append(block)

    # H/2 x W/2 -> H x W
    # ------------------------------------------------------------
    with tf.name_scope('block_6'):
        block = ReluDeconv2D('u3_0', block, 64, stride=2, size=4)
        block = ReluConv2D('u3_1', block, 64)
        block = ReluConv2D('u3_2', block, 64)
        block = ReluConv2D('u3_3', block, 6)
        block = ReluConv2D('u3_4', block, 3)
    estimate = tf.add(estimate, block, name='skip03')

    return estimate, skip_temporal_out
```

Glossary

Approximate Nearest Neighbor (ANN) The Approximate Nearest Neighbor is a vector v similar to a query q without any guarantee being the closest vector in the search space. This is acceptable in some applications to balance the tradeoff between accuracy and speed.

Angle Of Incidence (AOI) The Angle Of Incidence (AOI) is the angle between a ray incident on a surface and the surface normal at the point of incidence.

Convolutional Neural Network (CNN) A Convolutional Neural Network (CNN) is artificial deep neural network. In contrast to traditional deep artificial neural networks CNNs employ sparse connectivity and weight sharing.

Central Processing Unit (CPU) .

Dynamic Range Manipulation (DR) The Dynamic Range Manipulation (DR) is the proposed approach to synthesize high-dynamic range images from quantized 8bit RGB images..

frames per second (fps) .

Generative Adversarial Network (GAN) A Generative Adversarial Network (GAN) is a system of two artificial neural networks competing between synthesizing realistic samples (generator) and distinguishing these synthetic examples from real-world cases (discriminator/critic), see [Goo+14].

Graphics Processing Unit (GPU) .

Harris Corner Feature Detector (Harris) Harris Corner Feature Detector (Harris) is a popular feature detection algorithm in traditional computer vision [HS88].

Hierarchical Data Format 5 (HDF5) .

independent and identically distributed (i.i.d.) .

Local Curvature Generation (LCG) The Local Curvature Generation (LCG) is the proposed approach to synthesize reflections on non-flat surface.

Lightning Memory-Mapped Database (LMDB) .

Mutual Information (MI) Mutual Information (MI) between two random variables X and Y measures the amount of information about X provided by Y and vice versa, see Eq. (1.4) in [CT06]..

Non-rigid Deformation (NRD) The Non-rigid Deformation (NRD) is the proposed approach to synthesize miss-alignment between consecutive observation causes by moving objects in the scene..

Product Quantization Tree (PQT) A Product Quantization Tree is a hierachical extension of Product Quantization (PQ) reducing the number of full comparison when querying an ANN.

Product Quantization (PQ) Product Quantization is similar to Vector Quantization but uses codebooks for parts of a vector..

Point-Spread Function (PSF) The Point-Spread Function (PSF) represents the reponse of imaging system to a point light source. They can be directly observed as light streaks in images caused by point light sources and camera shakes..

Peak signal-to-noise ratio (PSNR) The Peak signal-to-noise ratio (PSNR) measures the ration between the signal and corruption by noise. When computing the PSNR between two images, the PSNR is a scaled version of the logarithm of the pixel-wise mean-squared error..

Recurrent Deblur Network (RDN) The Recurrent Deblur Network (RDN) is a specific network architecture introduced in this thesis which can handle streams of data of arbitrary sizes..

Recall at k (R@k) Recall at k ($R@k$) is the fraction of the first k retrieved elements that are relevant to a query q.

Scale-invariant Feature Transform (SIFT) Scale-invariant Feature Transform (SIFT) is a popular feature detection algorithm in traditional computer vision [Low04].

Simultaneous Localization and Mapping (SLAM) Simultaneous Localization and Mapping (SLAM) is a method to constructing or refining a virtual map of a real environment including the agent position.

Urban Reflection Database (URD) The Urban Reflection Database (URD) is a dataset captured to benchmark reflection and transmission separation algorithms.

Vector Quantization (VQ) Vector Quantization uses a codebook of vectors and compresses a vector by the id of the codebook entry which is closest to the vector.

with respect to (w.r.t.) .

Zero-Em-Queue (ZMQ) ZeroMQ (ZMQ) is an open-source universal high-speed messaging library..

Bibliography

[AB19] Karen Simonyan Andrew Brock Jeff Donahue. „Large Scale GAN Training for High Fidelity Natural Image Synthesis“. In: under review. 2019. URL: https://openreview.net/forums?id=B1xsqj09Fm.

[Aba+15] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[ADAS17] Nikolaos Arvanitopoulos Darginis, Radhakrishna Achanta, and Sabine Süsstrunk. „Single Image Reflection Suppression“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Aga+05] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David Salesin, and Richard Szeliski. „Panoramic Video Textures“. In: *ACM Transactions on Graphics (SIGGRAPH)*. SIGGRAPH '05. Los Angeles, California: ACM, 2005, pp. 821–827. DOI: 10.1145/1186822.1073268.

[AI08] Alexandr Andoni and Piotr Indyk. „Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions“. In: *Commun. ACM* 51.1 (Jan. 2008), p. 117. ISSN: 00010782. DOI: 10.1145/1327452.1327494. URL: http://portal.acm.org/citation.cfm?doid=1327452.1327494.

[Ara+16] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. „NetVLAD: CNN architecture for weakly supervised place recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Bar+09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. „PatchMatch: a randomized correspondence algorithm for structural image editing“. In: *ACM Transactions on Graphics (ToG)* 28.3 (2009), p. 24.

[BBK10] Alexander M. Bronstein, Michael M. Bronstein, and Ron Kimmel. „The Video Genome“. In: *CoRR* abs/1003.5320 (2010).

[BL12] Artem Babenko and Victor S. Lempitsky. „The inverted multi-index“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 3069–3076. ISBN: 978-1-4673-1226-4.

[Boj+16] Mariusz Bojarski et al. „End to End Learning for Self-Driving Cars“. In: *CoRR* abs/1604.07316 (2016).

[Bro+05] Alexander M Bronstein, Michael M Bronstein, Michael Zibulevsky, and Yehoshua Y Zeevi. „Sparse ICA for blind separation of transmitted and reflected images“. In: *International Journal of Imaging Systems and Technology* (2005).

[Bro+94] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. „Signature Verification using a "Siamese" Time Delay Neural Network“. In: *Advances in Neural Information Processing Systems (NIPS)*. 1994.

[Bro92] Lisa Gottesfeld Brown. „A Survey of Image Registration Techniques.“ In: *ACM Comput. Surv.* 24.4 (1992), pp. 325–376.

[BSH16] Jean-Charles Bazin and Alexander Sorkine-Hornung. „ActionSnapping: Motion-based Video Synchronization“. In: ECCV'16. Berlin, Heidelberg: Springer-Verlag, 2016.

[BSM12] N.J. Bryan, P. Smaragdis, and G.J. Mysore. „Clustering and synchronizing multi-camera video via landmark cross-correlation“. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2012, pp. 2389–2392. DOI: 10.1109/ICASSP.2012.6288396.

[BYO+01] Allan Kardec Barros, Tsuyoshi Yamamura, Noboru Ohnishi, et al. „Separating Virtual and Real Objects Using Independent Component Analysis“. In: *IEICE Transactions on Information and Systems* (2001).

[Cai+09] Jian-Feng Cai, Hui Ji, Chaoqiang Liu, and Zuowei Shen. „Blind motion deblurring using multiple images“. In: *Journal of computational physics* 228.14 (2009), pp. 5057–5071.

[Cha16] Ayan Chakrabarti. „A Neural Approach to Blind Motion Deblurring“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 221–235.

[Che+08] Jia Chen, Lu Yuan, Chi-Keung Tang, and Long Quan. „Robust dual motion deblurring“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2008, pp. 1–8.

[Che+14] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. „cuDNN: Efficient Primitives for Deep Learning.“ In: *CoRR* abs/1410.0759 (2014).

[Che+16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. „InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets“. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 2172–2180.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. „Learning a Similarity Metric Discriminatively, with Application to Face Verification.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2005, pp. 539–546. ISBN: 0-7695-2372-2.

[Cho+14] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[CI02] Yaron Caspi and Michal Irani. „Spatio-temporal alignment of sequences“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), pp. 1409–1424.

[Col05] Edward Collett. *Field guide to polarization*. SPIE press Bellingham, 2005.

[CPS06] Kumar Chellapilla, Sidd Puri, and Patrice Simard. „High performance convolutional neural networks for document processing“. In: 2006.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[Dat+04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. „Locality-sensitive Hashing Scheme Based on P-stable Distributions“. In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. New York, NY, USA: ACM, 2004, pp. 253–262. ISBN: 1-58113-885-7. DOI: 10.1145/997817.997857.

[Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. „ImageNet: A Large-Scale Hierarchical Image Database“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization“. In: *J. Mach. Learn. Res.* 12 (July 2011), pp. 2121–2159. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068.

[Die+11] Ferran Diego, Daniel Ponsa, Joan Serrat, and Antonio M. López. „Video Alignment for Change Detection.“ In: *IEEE Trans. Image Processing* 20.7 (2011), pp. 1858–1869.

[Dij59] E. W. Dijkstra. „A Note on Two Problems in Connexion with Graphs“. In: *NUMERISCHE MATHEMATIK* 1.1 (1959), pp. 269–271.

[Dos+15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. „Flownet: Learning optical flow with convolutional networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2758–2766.

[Dou+16] Matthijs Douze, Jérôme Revaud, Jakob Verbeek, Hervé Jégou, and Cordelia Schmid. „Circulant temporal encoding for video retrieval and temporal alignment“. In: *International Journal of Computer Vision* (2016). URL: https://hal.inria.fr/hal-01162603.

[Dre62] Stuart Dreyfus. „The numerical solution of variational problems“. In: *Journal of Mathematical Analysis and Applications* 5.1 (1962), pp. 30–45. ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(62)90004-5. URL: http://www.sciencedirect.com/science/article/pii/0022247X62900045.

[DS08] Yaron Diamant and Yoav Y Schechner. „Overcoming visual reverberations“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008.

[DS15a] Mauricio Delbracio and Guillermo Sapiro. „Burst deblurring: Removing camera shake through fourier burst accumulation.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2385–2393.

[DS15b] Mauricio Delbracio and Guillermo Sapiro. „Hand-Held Video Deblurring Via Efficient Fourier Aggregation“. In: *IEEE Transactions on Computational Imaging* 1.4 (2015), pp. 270–283.

[EB11] Georgios D. Evangelidis and Christian Bauckhage. „Efficient and Robust Alignment of Unsynchronized Video Sequences.“ In: *DAGM-Symposium*. Ed. by Rudolf Mester and Michael Felsberg. Vol. 6835. Lecture Notes in Computer Science. Springer, 2011, pp. 286–295. ISBN: 978-3-642-23122-3.

[EB13] Georgios D. Evangelidis and Christian Bauckhage. „Efficient Subframe Video Alignment Using Short Descriptors.“ In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.10 (2013), pp. 2371–2386.

[Eve+] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*.

[FA99] Hany Farid and Edward H Adelson. „Separating reflections and lighting using independent components analysis“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1999.

[Fan+17a] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. „A Generic Deep Architecture for Single Image Reflection Removal and Image Smoothing“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Fan+17b] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. „Revisiting Deep Intrinsic Image Decompositions“. In: *arXiv preprint arXiv: 1701.02965* (2017).

[FBF77] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. „An Algorithm for Finding Best Matches in Logarithmic Expected Time“. In: *ACM Trans. Math. Softw.* 3.3 (Sept. 1977), pp. 209–226. ISSN: 0098-3500. DOI: 10.1145/355744.355745.

[GCM14] Xiaojie Guo, Xiaochun Cao, and Yi Ma. „Robust Separation of Reflection from Multiple Images“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[Ge+13] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. „Optimized Product Quantization for Approximate Nearest Neighbor Search“. In: *CVPR 2013*. IEEE Computer Society, 2013.

[GO09] Michelle R. Greene and Aude Oliva. „The Briefest of Glances: The Time Course of Natural Scene Understanding“. In: *Psychological Science* 20.4 (2009), pp. 464–472.

[Goo+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. „Generative Adversarial Nets“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2672–2680.

[Gps] „Global Positioning System Standard Sositioning Service Serformance Standard“. In: (2008).

[GWL18] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. „Flex-Convolution (Deep Learning Beyond Grid-Worlds)“. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2018.

[HA14] Elad Hoffer and Nir Ailon. „Deep metric learning using Triplet network.“ In: *CoRR* abs/1412.6622 (2014).

[Has+09] Samuel W Hasinoff, Kiriakos N Kutulakos, Frédo Durand, and William T Freeman. „Time-constrained photography“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2009, pp. 333–340.

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition“. In: *arXiv preprint arXiv: 1512.03385* (2015).

[He+16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[He+16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[He+16c] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Identity Mappings in Deep Residual Networks“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2016, pp. 630–645.

[Hir+10] Michael Hirsch, Suvrit Sra, Bernhard Schölkopf, and Stefan Harmeling. „Efficient filter flow for space-variant multiframe blind deconvolution“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 607–614.

[Hra+15] Michal Hradiš, Jan Kotera, Pavel Zemcík, and Filip Šroubek. „Convolutional Neural Networks for Direct Text Deblurring“. In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2015, pp. 2015–10.

[HS17] Byeong-Ju Han and Jae-Young Sim. „Reflection Removal Using Low-Rank Matrix Completion“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[HS80] Berthold K.P. Horn and Brian G. Schunck. *Determining Optical Flow*. Tech. rep. Cambridge, MA, USA, 1980.

[HS88] Chris Harris and Mike Stephens. „A combined corner and edge detector“. In: *In Proc. of Fourth Alvey Vision Conference*. 1988, pp. 147–151.

[Hsu+14] Wei-Ning Hsu et al. „Hierarchical Generative Modeling for Controllable Speech Synthesis“. In: *arXiv preprint arXiv: 1810.07217* (2014).

[Hua+17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. „Densely connected convolutional networks“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Hua+18] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. „Multimodal Unsupervised Image-to-image Translation“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. „Distilling the Knowledge in a Neural Network“. In: *NIPS Deep Learning and Representation Learning Workshop*. 2015.

[HZ06] William W. Hager and Hongchao Zhang. „Algorithm 851: CG-DESCENT, a Conjugate Gradient Method with Guaranteed Descent“. In: *ACM Trans. Math. Softw.* 32.1 (2006), pp. 113–137. ISSN: 0098-3500. DOI: 10.1145/1132973.1132979.

[Ilg+17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. „Flownet 2.0: Evolution of optical flow estimation with deep networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2017, p. 6.

[IM98] Piotr Indyk and Rajeev Motwani. „Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality“. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Dallas, Texas, USA: ACM, 1998, pp. 604–613. ISBN: 0-89791-962-9. DOI: 10.1145/276698.276876.

[IS15] Sergey Ioffe and Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift“. In: *International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.

[Iso+17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. „Image-to-Image Translation with Conditional Adversarial Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Ito+14] ATSUSHI Ito, ASWIN C Sankaranarayanan, ASHOK Veeraraghavan, and RICHARD G Baraniuk. „Blurburst: Removing blur due to camera shake using multiple images“. In: *ACM Trans. Graph.* 3.1 (2014).

[Jan+17] Joel Janai, Fatma Güney, Jonas Wulff, Michael Black, and Andreas Geiger. „Slow Flow: Exploiting High-Speed Cameras for Accurate and Diverse Optical Flow Reference Data“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Piscataway, NJ, USA: IEEE, July 2017, pp. 1406–1416.

[JDJ17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. „Billion-scale similarity search with GPUs“. In: *arXiv preprint arXiv: 1702.08734* (2017).

[JDS11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. „Product Quantization for Nearest Neighbor Search.“ In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 33.1 (2011), pp. 117–128.

[Jég+11] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. „Searching in one billion vectors: re-rank with source coding“. In: *CoRR* abs/1102.3828 (2011).

[KA11] Simon Korman and Shai Avidan. „Coherency Sensitive Hashing“. In: *2011 Int. Conf. Comput. Vis.* (Nov. 2011), pp. 1607–1614. DOI: 10.1109/ICCV.2011.6126421. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126421.

[KA14] Y. Kalantidis and Y. Avrithis. „Locally Optimized Product Quantization for Approximate Nearest Neighbor Search“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Columbus, Ohio: IEEE, 2014.

[Kal60] Rudolph Emil Kalman. „A New Approach to Linear Filtering and Prediction Problems“. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[Kan+03] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. „High Dynamic Range Video“. In: *ACM Trans. Graph.* 22.3 (July 2003), pp. 319–325. ISSN: 0730-0301. DOI: 10.1145/882262.882270.

[Kar+18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. „Progressive Growing of GANs for Improved Quality, Stability, and Variation“. In: *International Conference on Learning Representations (ICLR).* 2018. URL: https://openreview.net/forum?id=Hk99zCeAb.

[KB14a] Diederik P. Kingma and Jimmy Ba. „Adam: A Method for Stochastic Optimization.“ In: *arXiv preprint arXiv: 1412.6980* (2014).

[KB14b] Diederik P. Kingma and Jimmy Ba. „Adam: A Method for Stochastic Optimization“. In: *International Conference on Learning Representations (ICLR)* (2014).

[KH96] D. Kundur and D. Hatzinakos. „Blind image deconvolution“. In: *IEEE Signal Processing Magazine* 13.3 (May 1996), pp. 43–64. ISSN: 1053-5888. DOI: 10.1109/79.489268.

[KNL16] Tae Hyun Kim, Seungjun Nah, and Kyoung Mu Lee. „Dynamic Scene Deblurring using a Locally Adaptive Linear Blur Model“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2016).

[Koc+06] Kristin Koch, Judith McLean, Ronen Segev, Michael A Freed, Michael J Berry, Vijay Balasubramanian, and Peter Sterling. „How Much the Eye Tells the Brain“. In: 16 (Aug. 2006), pp. 1428–34.

[KS99] Dong Sik Kim and Ness B. Shroff. „Quantization based on a novel sample-adaptive product quantizer (SAPQ).“ In: *IEEE Transactions on Information Theory* 45.7 (1999), pp. 2306–2320.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105.

[KTS] Naejin Kong, Yu-Wing Tai, and Joseph S Shin. „A physically-based approach to reflection separation: From physical modeling to constrained optimization“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* ().

[KW+52] Jack Kiefer, Jacob Wolfowitz, et al. „Stochastic estimation of the maximum of a regression function“. In: *The Annals of Mathematical Statistics* 23.3 (1952), pp. 462–466.

[KZ13] Ran Kaftory and Yehoshua Y Zeevi. „Blind separation of time/position varying mixtures“. In: *IEEE Transactions on Image Processing (TIP)* (2013).

[Köh+12] Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan Harmeling. „Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 27–40.

[Lai+17] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. „Deep Laplacian Pyramid Networks for Fast and Accurate Super-Resolution“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Lar+16] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. „Autoencoding Beyond Pixels Using a Learned Similarity Metric“. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. International Conference on Machine Learning (ICML). New York, NY, USA: JMLR.org, 2016, pp. 1558–1566. URL: `http://dl.acm.org/citation.cfm?id=3045390.3045555`.

[Las90] AN Lasenby. „Phase retrieval using HST images“. In: *Proceedings of the workshop at the Space Telescope Science Institute, Baltimore, USA*. Citeseer. 1990.

[Lau+16] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio. „Batch normalized recurrent neural networks“. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[LB13] Yu Li and Michael S Brown. „Exploiting reflection change for automatic reflection removal“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2013.

[LB14] Yu Li and Michael S Brown. „Single image layer separation using relative smoothness“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[LBG80] Y. Linde, A. Buzo, and R. M. Gray. „An Algorithm for Vector Quantizer Design“. In: *IEEE Transactions on Communications* 28 (1980), pp. 84–95.

[LeC+95] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, E Sackinger, Patrice Simard, et al. „Learning algorithms for classification: A comparison on handwritten digit recognition“. In: *Neural networks: the statistical mechanics perspective* 261 (1995), p. 276.

[Lev+09] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. „Understanding and evaluating blind deconvolution algorithms“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 1964–1971. DOI: `10.1109/CVPR.2009.5206815`.

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. „Microsoft COCO: Common Objects in Context“. In: *arXiv preprint arXiv: 1405.0312* (2014).

[Lin17] D. J. Lindler. „Block Iterative Restoration of Astronomical Images from the Hubble Space Telescope“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Lin70] S. Linnainmaa. „The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors“. In: (1970).

[Liu+18] 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)*. 2018.

[Llo06] S. Lloyd. „Least Squares Quantization in PCM“. In: *IEEE Trans. Inf. Theor.* 28.2 (Sept. 2006), pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489.

[Low04] David G. Lowe. „Distinctive Image Features from Scale-Invariant Keypoints“. In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[LW07] Anat Levin and Yair Weiss. „User assisted separation of reflections from a single image using a sparsity prior“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2007).

[LY07] Cheng Lei and Yee-Hong Yang. „Tri-focal tensor-based multiple video synchronization with subframe optimization.“ In: *IEEE Transactions on Image Processing* 15.9 (Mar. 22, 2007), pp. 2473–2480.

[LYT11] Ce Liu, J. Yuen, and A. Torralba. „SIFT Flow: Dense Correspondence across Scenes and Its Applications“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 33.5 (2011), pp. 978–994. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.147.

[MH08] Laurens van der Maaten and Geoffrey Hinton. „Visualizing data using t-SNE“. In: *Journal of Machine Learning Research* 9.Nov (2008), pp. 2579–2605.

[MI14] Tomer Michaeli and Michal Irani. „Blind deblurring using internal patch recurrence“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 783–798.

[Mit97] Thomas M. Mitchell. *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN: 0070428077, 9780070428072.

[ML14] Marius Muja and David G. Lowe. „Scalable Nearest Neighbor Algorithms for High Dimensional Data“. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36 (2014).

[MMPN16] Issey Masuda Mora, Santiago de la Puente, and Xavier Giro-i Nieto. „Open-ended visual question answering“. B.S. thesis. Universitat Politècnica de Catalunya, 2016.

[MSY16a] Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. „Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[MSY16b] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. „Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections“. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 2802–2810.

[MT94] Jorge J. Moré and David J. Thuente. „Line Search Algorithms with Guaranteed Sufficient Decrease“. In: *ACM Trans. Math. Softw.* 20.3 (1994), pp. 286–307. ISSN: 0098-3500. DOI: 10.1145/192115.192132.

[NCF17] Mehdi Noroozi, Paramanand Chandramouli, and Paolo Favaro. „Motion Deblurring in the Wild“. In: *arXiv preprint arXiv: 1701.01486* (2017).

[NH10] Vinod Nair and Geoffrey E Hinton. „Rectified linear units improve restricted boltzmann machines“. In: *International Conference on Machine Learning (ICML)*. 2010, pp. 807–814.

[NMZ05] Chong-Wah Ngo, Yu-Fei Ma, and Hong-Jiang Zhang. „Video summarization and scene detection by graph modeling“. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 15.2 (2005), pp. 296–305. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2004.841694.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.

[ODO16] Augustus Odena, Vincent Dumoulin, and Chris Olah. „Deconvolution and Checkerboard Artifacts“. In: *Distill* (2016). DOI: 10.23915/distill.00003. URL: http://distill.pub/2016/deconv-checkerboard.

[PHC16] Viorica Pătrăucean, Ankur Handa, and Roberto Cipolla. „Spatio-temporal video autoencoder with differentiable memory“. In: *International Conference on Learning Representations (ICLR) Workshop*. 2016.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. „U-Net: Convolutional Networks for Biomedical Image Segmentation“. In: *arXiv preprint arXiv: 1505.04597* (2015).

[RHW88] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. „Neurocomputing: Foundations of Research“. In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press, 1988. Chap. Learning Representations by Back-propagating Errors, pp. 696–699. ISBN: 0-262-01097-6. URL: http://dl.acm.org/citation.cfm?id=65669.104451.

[Ric+16] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. „Playing for Data: Ground Truth from Computer Games“. In: *European Conference on Computer Vision (ECCV)*. Ed. by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Vol. 9906. LNCS. Springer International Publishing, 2016, pp. 102–118.

[RM85] Herbert Robbins and Sutton Monro. „A stochastic approximation method“. In: *Herbert Robbins Selected Papers*. Springer, 1985, pp. 102–109.

[RW15] Dan Rosenbaum and Yair Weiss. „The Return of the Gating Network: Combining Generative Models and Discriminative Training in Natural Image Priors“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 2665–2673.

[Rüe+13] Jan Rüegg, Oliver Wang, Aljoscha Smolic, and Markus Gross. „DuctTake: Spatiotemporal Video Compositing“. In: *Computer Graphics Forum*. Vol. 32. 2pt1. Wiley Online Library. 2013, pp. 51–61.

[SAA00] Richard Szeliski, Shai Avidan, and P Anandan. „Layer extraction from multiple images containing reflections and transparency“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2000.

[Sch+13] Christian Schuler, Harold Burger, Stefan Harmeling, and Bernhard Scholköpf. „A machine learning approach for non-blind image deconvolution“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 1067–1074.

[Sch+15] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. „Learning to Deblur“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2015).

[Sch+17] Kevin Schawinski, Ce Zhang, Hantian Zhang, Lucas Fowler, and Gokula Krishnan Santhanam. „Generative adversarial networks recover features in astrophysical images of galaxies beyond the deconvolution limit“. In: *Monthly Notices of the Royal Astronomical Society: Letters* 467.1 (2017), pp. L110–L114.

[SGB19] Eli Schwartz, Raja Giryes, and Alexander M. Bronstein. „DeepISP: Toward Learning an End-to-End Image Processing Pipeline“. In: *IEEE Transactions on Image Processing (TIP)* 28.2 (2019), pp. 912–923.

[Shi+15] YiChang Shih, Dilip Krishnan, Fredo Durand, and William T Freeman. „Reflection removal using ghosting cues“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[SKS99] Yoav Y Schechner, Nahum Kiryati, and Joseph Shamir. „Separation of transparent layers by polarization analysis“. In: *Proceeding of the Scandinavian Conference on Image Analysis*. 1999.

[Spr+15] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. „Striving for Simplicity: The All Convolutional Net“. In: *arXiv:1412.6806, also appeared at ICLR 2015 Workshop Track*. 2015.

[SS02] Daniel Scharstein and Richard Szeliski. „A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms“. In: *International Journal of Computer Vision* 47.1 (2002), 7–42. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=64200`.

[SSK00] Yoav Y Schechner, Joseph Shamir, and Nahum Kiryati. „Polarization and statistical analysis of scenes containing a semireflector“. In: *Journal of the Optical Society of America* (2000).

[ST04] Peter Sand and Seth Teller. „Video Matching“. In: *ACM Trans. Graph.* 23.3 (Aug. 2004), pp. 592–599. ISSN: 0730-0301. DOI: `10.1145/1015706.1015765`.

[Su+17] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. „Deep Video Deblurring.“ In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[Sun+13] Libin Sun, Sunghyun Cho, Jue Wang, and James Hays. „Edge-based blur kernel estimation using patch priors“. In: *IEEE International Conference in Computational Photography (ICCP)*. IEEE. 2013, pp. 1–8.

[Sun+15] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. „Learning a convolutional neural network for non-uniform motion blur removal“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 769–777.

[Svo+16] Pavel Svoboda, Michal Hradis, Lukas Marsik, and Pavel Zemcik. „CNN for license plate motion deblurring“. In: *arXiv preprint arXiv: 1602.07873* (2016).

[SZ14] Karen Simonyan and Andrew Zisserman. „Very deep convolutional networks for large-scale image recognition“. In: *arXiv preprint arXiv: 1409.1556* (2014).

[Sze+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. „Going Deeper with Convolutions“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[TG04] Tinne Tuytelaars and Luc J. Van Gool. „Synchronizing Video Sequences“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2004, pp. 762–768. DOI: 10.1109/CVPR.2004.244. URL: http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.244.

[Tor+15] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. „24/7 place recognition by view synthesis“. In: *CVPR*. 2015.

[Tsa+14] Yun-Ta Tsai, Markus Steinberger, Dawid Pajak, and Kari Pulli. „Fast ANN for High-Quality Collaborative Filtering“. In: *High-Performance Graphics*. 2014, pp. 61–70.

[UI06] Yaron Ukrainitz and Michal Irani. „Aligning Sequences and Actions by Maximizing Space-time Correlations“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. ECCV'06. Graz, Austria: Springer-Verlag, 2006, pp. 538–550. ISBN: 3-540-33836-5, 978-3-540-33836-9. DOI: 10.1007/11744078_42.

[Vap92] V. Vapnik. „Principles of Risk Minimization for Learning Theory“. In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by J. E. Moody, S. J. Hanson, and R. P. Lippmann. Morgan-Kaufmann, 1992, pp. 831–838.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. „Attention is All you Need“. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 5998–6008.

[Vem04] Santosh Srinivas Vempala. *The Random Projection Method*. Vol. 65. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. DIMACS/AMS, 2004. ISBN: 0-8218-3793-1. URL: http://dimacs.rutgers.edu/Volumes/Vol65.html.

[VHW16] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. „Gated Siamese Convolutional Neural Network Architecture for Human Re-Identification.“ In: *CoRR* abs/1607.08378 (2016).

[Wad+18] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc Levoy. „Synthetic depth-of-field with a single-camera mobile phone“. In: *ACM Transactions on Graphics (ToG)* 37.4 (2018), p. 64.

[Wan+13] Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. „Regularization of Neural Networks using DropConnect.“ In: *International Conference on Machine Learning (ICML)*. 2013.

[Wan+14] Oliver Wang, Christopher Schroers, Henning Zimmer, Markus Gross, and Alexander Sorkine-Hornung. „VideoSnapping: Interactive Synchronization of Multiple Videos“. In: *ACM Trans. Graph.* 33.4 (July 2014), 77:1–77:10. ISSN: 0730-0301. DOI: 10.1145/2601097.2601208.

[Wan+17] Renjie Wan, Boxin Shi, Ling-Yu Duan, Ah-Hwee Tan, and Alex C Kot. „Benchmarking Single-Image Reflection Removal Algorithms“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Wer81] P. J. Werbos. „Applications of Advances in Nonlinear Sensitivity Analysis“. In: *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*. 1981, pp. 762–770.

[WFL17] Patrick Wieschollek, Ido Freeman, and Hendrik P. A. Lensch. „Learning Robust Video Synchronization without Annotations“. In: *IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dec. 2017. DOI: 10.1109/ICMLA.2017.0-173.

[WGL17] Patrick Wieschollek, Fabian Groh, and Hendrik P. A. Lensch. „Backpropagation Training for Fisher Vectors within Neural Networks“. In: *CoRR* (2017).

[Wie+16a] Patrick Wieschollek, Oliver Wang, Alexander Sorkine-Hornung, and Hendrik P.A. Lensch. „Efficient Large-scale Approximate Nearest Neighbor Search on the GPU“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016. DOI: 10.1109/CVPR.2016.223.

[Wie+16b] Patrick Wieschollek, Bernhard Schölkopf, Hendrik P. A. Lensch, and Michael Hirsch. „End-to-End Learning for Image Burst Deblurring“. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Nov. 2016. DOI: 10.1007/978-3-319-54190-7_3.

[Wie+16c] Patrick Wieschollek, Berhard Schölkopf, Hendrik P.A. Lensch, and Michael Hirsch. „End-to-End Learning for Image Burst Deblurring." In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2016.

[Wie+17] Patrick Wieschollek, Michael Hirsch, Bernhard Schölkopf, and Hendrik P. A. Lensch. „Learning Blind Motion Deblurring". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017. DOI: 10.1109/ICCV.2017.34.

[Wie+18] Patrick Wieschollek, Orazio Gallo, Jinwei Gu, and Jan Kautz. „Separating Reflection and Transmission Images in the Wild". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[Win80] Shmuel Winograd. *Arithmetic complexity of computations*. Vol. 33. Siam, 1980.

[WT14] Ruxin Wang and Dacheng Tao. „Recent progress in image deblurring". In: *arXiv preprint arXiv: 1409.6838* (2014).

[WZ06] Lior Wolf and Assaf Zomet. „Wide Baseline Matching between Unsynchronized Video Sequences". In: *International Journal of Computer Vision (IJCV)* 68.1 (2006), pp. 43–52. DOI: 10.1007/s11263-005-4841-0.

[Xu+14] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. „Deep Convolutional Neural Network for Image Deconvolution". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. 2014, pp. 1790–1798.

[Xue+15] Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. „A computational approach for obstruction-free photography". In: *ACM Transactions on Graphics (SIGGRAPH)* (2015).

[ZC14] Haichao Zhang and Lawrence Carin. „Multi-shot imaging: joint alignment, deblurring and resolution-enhancement". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 2925–2932.

[ZD09] Feng Zhou and Fernando De la Torre Frade. „Canonical Time Warping for Alignment of Human Behavior". In: *Advances in Neural Information Processing Systems (NIPS)*. 2009.

[Zho+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. „Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2017).

[Zhu+17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. „Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networkss“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[ZJH19] Ningyuan Zheng, Yifan Jiang, and Dingjiang Huang. „StrokeNet: A Neural Painting Environment“. In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=HJxwDiActX.

[ZK15] Sergey Zagoruyko and Nikos Komodakis. „Learning to Compare Image Patches via Convolutional Neural Networks“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[ZKL18] Jialiang Zhang, Soroosh Khoram, and Jing Li. „Efficient Large-Scale Approximate Nearest Neighbor Search on OpenCL FPGA“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4924–4932.

[ZW11] D. Zoran and Y. Weiss. „From learning models of natural image patches to whole image restoration“. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Nov. 2011, pp. 479–486. DOI: 10.1109/ICCV.2011.6126278.

[ZWZ13] Haichao Zhang, David Wipf, and Yanning Zhang. „Multi-image blind deblurring using a coupled adaptive sparse prior“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 1051–1058.

[ZWZ14] Haichao Zhang, David P. Wipf, and Yanning Zhang. „Multi-Observation Blind Deconvolution with an Adaptive Sparse Prior“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 36.8 (2014), pp. 1628–1643. DOI: 10.1109/TPAMI.2013.241.

[ZY15] Haichao Zhang and Jianchao Yang. „Intra-Frame Deblurring by Leveraging Inter-Frame Camera Motion“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4036–4044.

[ZŠM12] Xiang Zhu, Filip Šroubek, and Peyman Milanfar. „Deconvolving psfs for a better motion deblurring using multiple images“. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 636–647.

[Ble17] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Blender Institute, Amsterdam, 2017.

[Whi92] R. L. White. „Restoration of Images and Spectra from the Hubble Space Telescope“. In: *Astronomical Data Analysis Software and Systems*. Ed. by D. M. Worrall, C. Biemesderfer, and J. Barnes. Vol. 25. Astronomical Society of the Pacific Conference Series. 1992, p. 176.

[ŠM12] Filip Šroubek and Peyman Milanfar. „Robust multichannel blind deconvolution via fast alternating minimization“. In: *Image Processing, IEEE Transactions on* 21.4 (2012), pp. 1687–1700.

Contributions

Except otherwise explicitly stated, all mathematical derivations, algorithmic implementations and experimental evaluations were performed by the author of this thesis. Experimental evaluations of cited methods have either been produced by the authors of the respective method or with implementations of the method provided by the authors.

All figures and photographs were captured by the author of this thesis if not stated otherwise. The video material for Chapter 4 has been recorded by Hendrik Lensch.

The result in Figure 4.18 was jointly produced with Veronika Kohler during supervising her master's thesis.