IJCV VISI manuscript No.
(will be inserted by the editor)

What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?

Nikolaus Mayer · Eddy Ilg · Philipp Fischer · Caner Hazirbas · Daniel Cremers · Alexey Dosovitskiy · Thomas Brox

Received: date / Accepted: date

Abstract The finding that very large networks can be trained efficiently and reliably has led to a paradigm shift in computer vision from engineered solutions to learning formulations. As a result, the research challenge shifts from devising algorithms to creating suitable and abundant training data for supervised learning. How to efficiently create such training data? The dominant data acquisition method in visual recognition is based on web data and manual annotation. Yet, for many computer vision problems, such as stereo or optical flow estimation, this approach is not feasible because humans cannot manually enter a pixel-accurate flow field. In this paper, we promote the use of synthetically generated data for the purpose of training deep networks on such tasks. We suggest multiple ways to generate such data and evaluate the influence of dataset properties on the performance and generalization properties of the resulting networks. We also demonstrate the benefit of learning schedules that use different types of data at selected stages of the training process.

Keywords Deep Learning · Data Generation · Synthetic Ground Truth · FlowNet · DispNet

We acknowledge funding by the ERC Starting Grant VideoLearn, the ERC Consolidator Grant "3D Reloaded", the DFG Grant BR-3815/7-1, the DFG Grant CR 250/17-1, and the EU Horizon2020 project TrimBot2020. We thank Benjamin Ummenhofer for code that kick-started the creation of our 3D datasets.

Nikolaus Mayer · Eddy Ilg · Philipp Fischer · Alexey Dosovitskiy · Thomas Brox
University of Freiburg
E-mail: {mayern,ilg,fischer,dosovits,brox}@cs.uni-freiburg.de

Caner Hazirbas · Daniel Cremers
Technical University of Munich
E-mail: {hazirbas,cremers}@in.tum.de

1 Introduction

Since the beginning of research in artificial intelligence, there has been a constant shift from engineering of solutions for special scenarios towards methodologies that are more generally applicable. The increased popularity of learning methods, which already started before the success of deep learning, is another chapter in this progress towards more generic approaches. These methods allow restricting the engineering on the algorithmic side, but at the same time much more influence is given to the data. While classical computer vision methods have not required any training data, in which sense they are unsupervised methods[1], the final performance of approaches following the dominant supervised learning paradigm depends very much on the size and quality of training datasets. Especially the large potential of deep learning can only be fully exploited with large datasets.

Significant research efforts have been made to create such datasets, *e.g.* ImageNet (Deng et al, 2009), MS COCO (Lin et al, 2014), CityScapes (Cordts et al, 2016), or the NYU dataset (Silberman et al, 2012). These datasets have enabled most of the progress in computer vision in recent years. Notably, all these datasets cover the field of visual recognition such as object detection and classification. This has been one of the reasons why deep learning in computer vision has mainly focused on visual recognition for a long time. A notable exception is the NYU dataset. It contains a large collection of RGB-D images and enabled the seminal work of Eigen et al (2014) on depth prediction from a single image.

[1] Clearly, the engineering and optimization of tuning parameters requires some training data. For a long time, test data was misused for this purpose.

In this paper, we present and analyze one of the very first approaches to create training data for learning optical flow and disparity estimation—two classical computer vision tasks—on a large scale. Creating data for such tasks requires a different approach than in visual recognition, where web data in conjunction with manual annotation by mostly untrained persons can yield large datasets with reasonable time and monetary effort. In case of optical flow estimation, for instance, it is not obvious how to derive ground truth for large sets of real videos. Notable works include the Middlebury dataset (Baker et al, 2011) and the KITTI datasets (Geiger et al, 2012; Menze and Geiger, 2015), but due to the difficulty of computing accurate ground truth, these have been restricted to 8 and $2 \cdot 200$ pairs of frames, respectively (the latter are also limited by the special driving setting). This is far from what is needed to train a powerful deep network. Therefore, these datasets have been used mainly as test sets for classical, non-learning-based methodology. The situation is similar for disparity estimation from stereo pairs, or depth estimation and camera tracking from monocular videos.

The MPI-Sintel dataset demonstrated the feasibility of using existing data from an open source movie to render videos together with the desired ground truth (Butler et al, 2012). Sintel provides ground truth for optical flow, disparities and occlusion areas, and has been largely adopted as a serious benchmark dataset despite its synthetic nature. Still, the dataset is small by deep learning standards: The training set consists of 1041 image pairs – which is sufficient to train a network, but not a very powerful one, as we show in this paper. Moreover, the approach does not scale arbitrarily due to the limited availability of open source movies.

Inspired by the Sintel dataset, we take the approach of rendering images together with various ground truth outputs to a new level. Embracing procedural generation and abstract instead of naturalistic data, we focus on training rather than testing and aim for a much larger scale. In contrast to the Sintel dataset, which was intended for benchmarking, this allows us to ignore many subtle problems that appear in the rendering of synthetic data (Wulff et al, 2012), and rather focus on the size and diversity of the dataset.

There are many ways to generate training data in a synthetic manner: using existing scene data as in Sintel, manually designing new scenes, or creating randomized scenes in a procedural manner. In this paper, we investigate which of these options are most powerful and which properties of a dataset are most important for training a network that will generalize also to other data, particularly real data. We focus this investigation on networks that are trained for optical flow estimation, specifically FlowNet (Dosovitskiy et al, 2015), and disparity estimation, specifically DispNet (Mayer et al, 2016).

The study leads to many interesting findings. For instance, the data does not have to be realistic to make for a good training dataset. In fact, our simplistic 2D FlyingChairs dataset yields good data to start training a network for optical flow estimation. Moreover, we find that:

- multistage training on multiple separate datasets works better than not only either one of the datasets by itself, but also than a full mix of both,
- enabling more realism in the data via complex lighting does not necessarily help even if the test data is realistically lit and
- simulating the flaws of a real camera during training improves network performance on images from such cameras.

2 Related Work

Synthetic, rendered data has been used for benchmarking purposes for a long time. Heeger (1987) and Barron et al (1994) used virtual scenes (including Lynn Quam's famous *Yosemite* 3D sequence) to quantitatively evaluate optical flow estimation methods. McCane et al (2001) composed images of objects onto backgrounds and varied the complexity of scenes and motions to quantify how these changes affect estimation accuracy. Complementary to these are the works of Otte and Nagel (1995) who recorded real scenes with simple geometry and camera motion such that the ground truth flow could be annotated manually, and Meister and Kondermann (2011) who acquired ground truth for a real scene by recreating it as a virtual 3D model. The Middlebury flow dataset (Baker et al, 2011) contains both real and synthetic scenes. Vaudrey et al (2008) created synthetic sequences for benchmarking scene flow estimation. Somewhere between synthetic and real are datasets such as Dwibedi et al (2017) who created collages from real photographs to get training data for instance detection.

Recent synthetic datasets include the UCL dataset by Baker et al (2011) and the larger Sintel benchmark by Butler et al (2012) where the 3D software *Blender* was used to obtain dense and accurate flow and disparity ground truth for complex rendered scenes. Onkarappa and Sappa (2014) used the comparable 3D software *Maya* to render a small driving-specific dataset for their study on optical flow accuracy. Table 1 gives an overview of datasets for optical flow and depth

Dataset	Published in	Synthetic/Natural	Flow	Depth	Stereo	#Frames	Resolution
Middlebury	Baker et al (2011)	S/N	✓			8	640 × 480
UCL	Baker et al (2011)	S	✓			20	640 × 480
KITTI 2012	Geiger et al (2012)	N	✓	✓	✓	194	1,242 × 375
NYU v2	Silberman et al (2012)	N		✓		408,473	640 × 480
Sintel	Butler et al (2012)	S	✓		✓	1,064	1,024 × 436
TUM	Sturm et al (2012)	N		✓		∼ 100,000	640 × 480
Sun3D	Xiao et al (2013)	N		✓		∼ 2,500,000	640 × 480
Middlebury 2014	Scharstein et al (2014)	N		✓		23	∼ 6MP
KITTI 2015	Menze and Geiger (2015)	S	✓	✓	✓	200	1,242 × 375
FlyingChairs	Dosovitskiy et al (2015)	S	✓			21,818	512 × 384
FlyingThings3D	Mayer et al (2016)	S	✓	✓	✓	22,872	960 × 540
Monkaa	Mayer et al (2016)	S	✓	✓	✓	8,591	960 × 540
Driving	Mayer et al (2016)	S	✓	✓	✓	4,392	960 × 540
Virtual KITTI	Gaidon et al (2016)	S	✓	✓		21,260	1,242 × 375
SYNTHIA	Ros et al (2016)	S		✓	✓	∼ 200,000	960 × 720
ScanNet	Dai et al (2017)	N		✓		∼ 2,500,000	640 × 480
SceneNet RGB-D	McCormac et al (2017)	S	✓	✓		∼ 5,000,000	320 × 240

Table 1 Datasets for optical flow and depth estimation. The multitude and variation among the datasets makes choosing one nontrivial. This overview is ordered by date of publication. Many datasets focus on specific settings or features.

estimation and their technical properties. Examples for most of them are shown in Fig. 1.

By hooking into the popular Unreal game engine, Qiu and Yuille (2016) enabled data generation using existing assets made by game content creators. Their technique was used by Zhang et al (2016) to make a disparity evaluation dataset for difficult visual effects such as specularity. The game engine approach was also taken by Taylor et al (2007) to build test data for video surveillance systems, as well as by Dosovitskiy et al (2017) to build the CARLA driving simulator. Our work uses the concept of synthetic data, but focuses on large scale training data rather than benchmark datasets. Moreover, we present multiple ways of creating such data: using an open source movie as in Butler et al (2012), manually building custom 3D scenes, and random procedural generation of scenes.

Several recent and concurrent works have focused on building large scale synthetic training datasets for various computer vision tasks. These datasets are typically specialized on narrow scenarios: automotive driving in the KITTI-lookalike Virtual KITTI dataset (Gaidon et al, 2016), the SYNTHIA dataset (Ros et al, 2016) and the dataset of Richter et al (2016); indoor scenes in SceneNet (Handa et al, 2016), SceneNet RGB-D (McCormac et al, 2017), ICL-NUIM (Handa et al, 2014) and SUNCG (Song et al, 2017); isolated objects in ModelNet (Wu et al, 2015) and ShapeNet (Chang et al, 2015); human action recognition in de Souza et al (2017). In contrast to these works, we are focusing on datasets which allow learning general-purpose optical flow and disparity estimation.

There has been relatively little work on analyzing which properties of synthetic training data help generalization to real data. Su et al (2015) used rendered data to train a network on object viewpoint estimation; their results indicate better performance in real scenes when the training data contains real background textures and varying lighting. Movshovitz-Attias et al (2016) investigated how lighting quality in rendered images affects the performance of a neural network on viewpoint estimation, and found that more realistic lighting helps. Zhang et al (2017) studied the influence of synthetic data quality on semantic segmentation. In this work, we perform an in-depth analysis of synthetic datasets for optical flow estimation, varying not only the lighting conditions, but also the shapes, motions and textures of the objects in the scene.

The present paper consolidates and extends three earlier conference papers (Dosovitskiy et al, 2015; Mayer et al, 2016; Ilg et al, 2017) with the focus on training data. In the conference papers, datasets with 2D shapes (FlyingChairs, Dosovitskiy et al (2015) and ChairsSDHom, Ilg et al (2017)) and more sophisticated datasets with 3D objects (Mayer et al, 2016) were introduced and used to train CNNs for optical flow and disparity estimation. In this work, we provide a principled study from simplistic to more sophisticated training data and analyze the effects of data properties on network training and generalization.

3 Synthetic Data

Supervised training of deep neural networks requires large amounts of data. For visual recognition, large training datasets can be acquired from web data with manual annotations, such as class labels, bounding boxes,

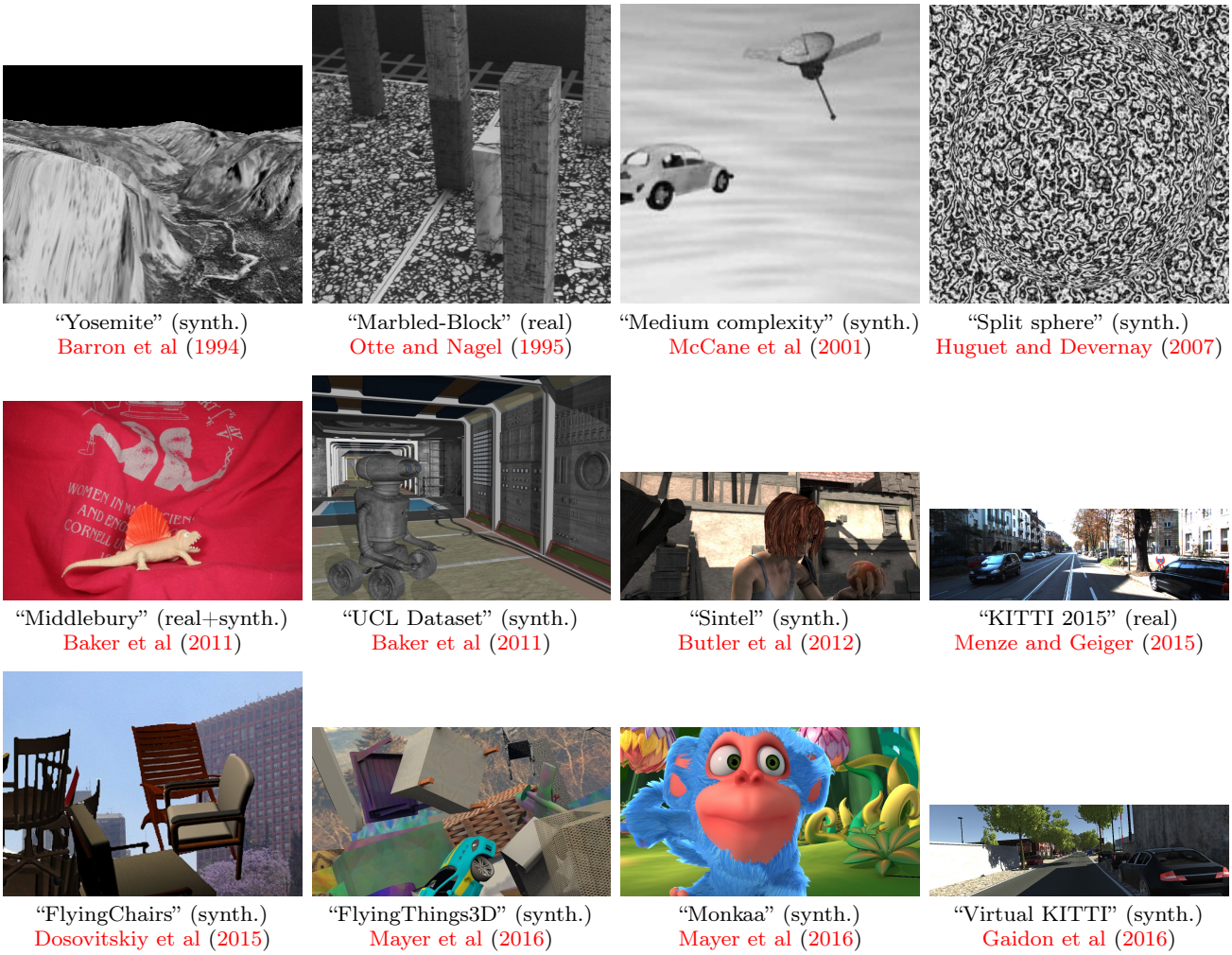


Fig. 1 Samples from real and synthetic datasets with optical flow annotation. Real datasets are small and restricted in diversity due to the difficulty to derive ground truth optical flow. Synthetic data can be based on manual scene modeling, such as the open-source Sintel and Monkaa movies, or procedural generation, such as the FlyingChairs and FlyingThings3D datasets.

or object outlines. For dense low-level tasks, such as optical flow or depth estimation, manual annotation is not feasible, especially when exact and pixel-accurate ground truth is desired. This motivates the use of synthetic data, *i.e.* data not taken from the real world but artificially generated: the key idea is that we have perfect knowledge and control of the virtual "world" from which we create our image data, and that we are thus able to produce perfect ground truth annotations along with the images. The difficulties are that creating photo-realistic image data is not trivial, and that it is not obvious which aspects of the real world are relevant and must be modeled (these problems arise because we are unable to perfectly simulate the real world).

This section recapitulates our prior works' discussions about synthetic training data generation and data augmentation. We describe two ways to generate synthetic data: procedural randomization and manual modeling. Using our randomized 2D and 3D datasets as examples, we first discuss how starting off with a simple general scene and combining it with randomly sampled dynamic elements can yield datasets that are abstract but powerful and potentially of unlimited size. The dynamic elements are motivated by what the dataset should be able to teach, *e.g.* optical flow due to object rotation.

Following this, we discuss manual data modeling in which more specialized datasets are created. This can be achieved either by rendering existing scenes (an approach used for the Sintel dataset of Butler et al (2012) and our own Monkaa dataset (Mayer et al, 2016)), or by manually recombining objects from existing scenes to get new, specially designed scenes[2]. We used the latter

[2] There is also the extreme case of handcrafting entire scenes and all their objects. We do not consider this option here because it yields even far less data for the same effort.

method in our Monkaa and Driving datasets. Mixtures of both methods are also possible, as demonstrated by Virtual KITTI where *e.g.* cars can be randomly modified or left out of a fixed scene (Gaidon et al, 2016). Video game engines allow tackling this from yet another perspective: using a game means offloading the model and scene design to its developers. The mechanics of the game world can then be used to generate large amounts of data, within the constraints of the game. This was used *e.g.* by Richter et al (2016), Richter et al (2017) and Dosovitskiy et al (2017). This approach is an alternative to ours and not considered in this paper.

3.1 Randomized modeling of data

3.1.1 2D image pairs for optical flow: FlyingChairs

Optical flow estimation is the task of taking two images which show basically the same content with some variation (*e.g.* moving objects or different camera poses) and determining the apparent pixel motion of each pixel from the first to the second image. It is a low-level, nonsemantic task; any two images between which correspondences can be determined is a valid input. This means that to learn this task, any image pair with computable pixelwise ground truth displacements can serve as training data.

A simple way of creating data with ground truth displacements is to take images of objects (segmented such that anything but the object itself is masked out) and paste them onto a background, with a randomized transformation applied from the first to the second image. Fig. 2 illustrates how a FlyingChairs sample is created this way. We used images of chairs created from CAD models by Aubry et al (2014). We chose chairs because: (a) flying chairs offer nowhere near the semantic content of real-world scenes (thus, successful generalization to real data indicates that the estimation of point correspondences can be learned from abstract data) and (b) chairs have non-convex shapes with fine structure and are topologically diverse.

We superimposed the chair images onto real-world background images of urban and landscape environments downloaded from Flickr. In more detail, to generate the first image of each sample pair, random affine transformations were applied to all objects, *i.e.* the chairs were randomly scaled, rotated, and placed on the randomly transformed background. To generate the second image, additional small random affine transformations were generated for all objects. These second transformations model the motion from the first to the second image. We simulate "camera motion" effects, *i.e.* that objects tend to move with the scene, by composing the background's motion onto each object's own motion. Since all transformation parameters are known, accurate ground truth optical flow is available at every single pixel, even in regions that become occluded in the second image. The resulting images resemble those from McCane et al (2001) (*cf.* Fig. 1), yet with the crucial difference that there are thousands of randomized scenes instead of a few manually created ones.

3.1.2 Synthetic 3D data for optical flow, disparity and scene flow: FlyingThings3D

The FlyingChairs were generated using simple 2D affine transformations. Since such datasets cannot contain information from depth in the scene, *i.e.* 3D rotation and translation in space and camera motion, we took the randomization approach further and created FlyingThings3D: a 3D dataset rendered from true 3D models, with ground truth for stereo disparity, optical flow, and for the full scene flow task (Mayer et al, 2016).

To obtain 3D scenes, we created a simple but structured background from simple geometric random shapes. For the dynamic foreground objects, we used models from ShapeNet (Chang et al, 2015). All foreground objects follow linear trajectories in 3D space, and so does the camera. To animate the camera's viewing direction, an invisible moving object is added to the scene and the camera is constrained to always point or "look" at this object. This 3D setting with combined object and camera motions allows for complex object flows and scene configurations.

To render the 3D models into 2D images, we used the freely available Blender suite. We modified its internal rendering engine to directly produce fully dense and accurate ground truth for depth, disparity, optical flow and object segmentation (among others), all for both views of a virtual stereo camera.

The FlyingThings3D dataset combines sophisticated 3D rendering with the procedural generation of scenes which allows for arbitrary amounts of data without manual effort. While the generated scenes are by no means realistic in the naturalistic sense of *e.g.* Sintel, they allow for a large and diverse dataset.

3.2 Manual modeling

In the previous section, we presented approaches to generate large amounts of data automatically. This section discusses the generation of synthetic data that involves manual engineering. We use the 3D model rendering approach described in Section 3.1.2.

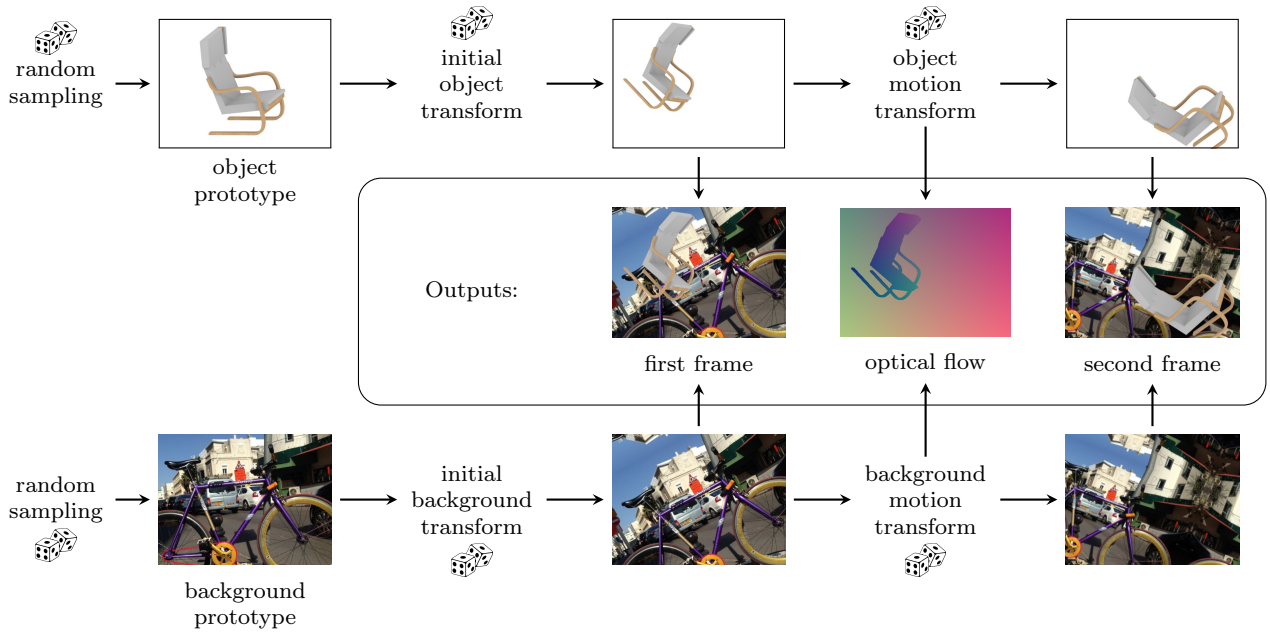


Fig. 2 FlyingChairs: A two-frame sample is created by composing a foreground object onto backgrounds. Each object and background has an initial transform (to introduce scene variety), as well as a transform between frames which induces optical flow. All transforms are affine which makes computing the ground truth flow field easy. This schematic shows a simplification for a single foreground object; in our datasets there are multiple objects, each with an individual transform. Note that the foreground object's transform is composed onto the one of the background, *i.e.* the object will move "with the background" if its own transform is the identity. This correlates object motion with background motion and simulates optical flow induced by camera motion. Dice icons indicate randomization.

3.2.1 Monkaa

The publicly available open source movie Monkaa provides 3D scenes and assets that can be loaded into Blender. We created a dataset that contains original scenes as well as new custom ones (Mayer et al, 2016).

In contrast to FlyingThings3D, data is generated from the movie in a deterministic way. The original scenes and objects were modeled by 3D artists. For our custom scenes, we manually collected, composed and animated original set pieces and objects from the movie, producing entirely new environments and movements while keeping the visual style of the Monkaa movie. To obtain a sufficient amount of data, we rendered longer scenes instead of procedurally generating many variations. Contrary to the datasets mentioned above, Monkaa contains articulated non-rigid motion of animals and extremely complex fur. Fig. 1 shows a frame from our Monkaa dataset release.

During our work on Monkaa, we encountered many of the problems described by the creators of the Sintel benchmark dataset (Wulff et al, 2012), such as changing focal length, scenes containing objects only in the exact locations viewed by the camera and optical tricks which break when using a stereo setup (*e.g.* forced perspective or "fog" rendered as a flat 2D sprite). This greatly limited the amount of usable scenes and contributes to the fact that this approach to data generation cannot be scaled easily to produce more data.

3.2.2 Driving

Videos captured from a camera on a driving car provide a very special setting and usually differ significantly from other video material. This is demonstrated well in the KITTI benchmark suite (Geiger et al, 2012). Wide-angle lenses are typically used to cover large areas, and the scene motion is dominated by the forward camera motion of the driving car. Most of the scene is static and the arrangement of objects in the scenes is very similar, with sky at the top, the road at the bottom and mostly walls and parked vehicles on the sides. These motion and object types are not covered in the previously presented data and hence require special treatment. To this end, we created a dataset that represents this setting and resembles the typical aspects of a real dataset like KITTI. The 3D scenes contain simple blocks and cylinders to imitate buildings, and models of cars, trees, and street lamps taken from 3D Warehouse[3] and the ShapeNet database. The stereo baseline, the focal

[3] `https://3dwarehouse.sketchup.com/`

length and properties such as scene lighting and object materials were chosen to match the KITTI dataset. The virtual camera and other cars in the scene were manually animated into a simple urban traffic scenario. While the dataset does not make much sense on a high level (there are no road signs, traffic light situations, or pedestrians crossing the road), it does replicate the rough semantic structure of a typical KITTI scene. The experiments in this paper confirm the intuition that fine-tuning on this data should teach a network useful priors, *e.g.* a flat road surface at the bottom, and thus largely improve the performance on KITTI compared to training on data without such priors.

3.3 Data augmentation

The term "augmentation" refers to artificially increasing the amount of data that is presented while training a machine learning algorithm, with the goal of improving generalization capabilities. This is done by transforming the original data in small, incremental ways. It relies on the assumption that the learning algorithm is unable to filter out the small changes and instead perceives the modified data as actual *new* data with new information. While our 2D and 3D datasets offer much more training data than had previously been available, the use of data augmentation within the training process is much more efficient: additional data need not be stored and read from disk during training but can be generated on the fly. We split our augmentation options into color and geometry augmentations. Our experiments indicate that the two types offer complementary benefits.

To train optical flow networks we used color augmentation (changing brightness, contrast and colors and adding color noise). Options for geometric changes here include every change for which we can easily and directly compute an accurate ground truth flow field: shift, rotation and scaling. We can simply apply the same transformation to both input images, or additionally apply a second transformation to only one of the images. The latter can produce a greater variety of augmentations, *e.g.* by inducing a scaling motion between frames. Note that any of the described geometry augmentations still yields a valid optical flow training sample by simply composing the computed flow induced by the augmentation operation with the existing flow of the original sample.

For our disparity networks, we use the same color augmentations as for optical flow. However, disparity estimation for stereo cameras is a more geometrically constrained setting than optical flow: the transformation between the two cameras is fixed, and *e.g.* rotating both views of a sample pair or zooming one view relative to the other would disturb the epipolar geometry. This means that for disparity the setting is more restricted and fewer augmentation options are available.

4 Learning Tasks

4.1 Optical Flow Estimation

The FlowNet was introduced by Dosovitskiy et al (2015). This network was the first to be trained end-to-end for optical flow estimation. It was designed to receive two images of the video in full resolution as input and generate the corresponding optical flow field as output.

In Dosovitskiy et al (2015), two architectures were proposed, FlowNetS and FlowNetC (see Fig. 3), which share a common underlying idea. The FlowNetS input consists of two RGB input images stacked into a six-channel input blob. The processing starts with a contracting part that compresses the spatial image information into feature blobs with lower resolution but more feature channels. The subsequent expanding part uses up-convolutions to successively increase the resolution until it reaches the desired output size. Skip-connections transfer higher resolution information directly from a resolution level of the contracting part to its corresponding resolution in the expanding part, thus this information does not have to pass through the architecture's bottleneck.

Each convolutional block consists of a convolution and a ReLU nonlinearity. The filter size decreases while the number of feature maps increases as shown in Fig. 3. For spatial reduction of resolution, each convolution of the contracting part is strided with a factor of 2. In the expanding part we use up-convolutions with stride 2 to increase the resolution again. After each expansion step, an endpoint error (EPE) loss aids the network with additional training gradients in a deep-supervision manner. For more details, we refer to Dosovitskiy et al (2015).

The FlowNetC variant of this architecture (shown in Fig. 3) employs a custom layer which computes the correlation scores between patches from two input feature streams. In this variant, the two input images are processed separately in the first three layers, but with shared weights, *i.e.* as a Siamese network. The resulting features are fused using the correlation layer and then processed further as in the FlowNetS.

In Ilg et al (2017), we found that the FlowNetC consistently outperforms the FlowNetS. Therefore, unless mentioned otherwise, all experiments on optical flow in this paper are based on the FlowNetC architecture. Specifically, we used "FlowNet2-c" (Ilg et al, 2017), a FlowNetC in which each layer's number of channels was

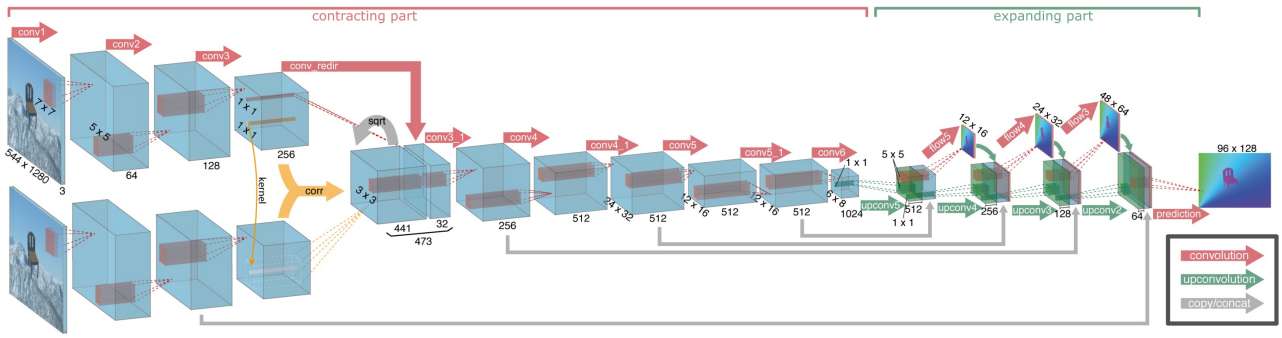


Fig. 3 FlowNetC architecture from Dosovitskiy et al (2015). The figure shows the full 15-layer architecture with contracting and expanding part. The gray skip connections between both parts allow for high-resolution predictions without passing high-resolution information through the bottleneck. We use the same architecture with fewer feature channels in our optical flow experiments. The DispNetCorr1D from Mayer et al (2016) (which we use for our disparity experiments in Section 5.4) implements the same idea, but with only a 1D correlation.

reduced to $\frac{3}{8}$ of the original size. The reduced computational cost during training allows for the large set of evaluations in this paper. Unless specifically mentioned, we trained for 600k mini-batch iterations with a batch size of 4, following the S_{short} learning rate schedule from Ilg et al (2017) (see also Fig. 8): starting with learning rate 1e−4, then successively dropping to 5.0e−5, 2.5e−5, and 1.25e−5 at 300k, 400k, and 500k iterations respectively. Our experiments in Section 5.3 use the S_{long} and S_{fine} schedules intended for pretraining and finetuning on different datasets.

4.2 Disparity Estimation

In our disparity experiments, we used the DispNet from Mayer et al (2016) in its DispNetCorr1D variant. Like FlowNetC, this network receives two images as input and initially processes them in two separate streams with shared weights (Siamese architecture). The features at the end of both streams are correlated in a correlation layer and jointly processed further. By exploiting the known and fixed epipolar geometry of rectified stereo pairs, the correlation is reduced to unidirectional 1D on horizontal scanlines. Thus, compared to a FlowNet with a 2D correlation layer, the DispNetCorr1D can process a much wider receptive field at the same computational cost.

5 Experiments

5.1 Testing environments

In this paper, we aim for the analysis of relevant dataset properties that make a good dataset for training large networks for optical flow and disparity estimation.

Training data	Test data: Sintel	KITTI2015	FlyingChairs
Sintel	6.42	18.13	5.49
FlyingChairs	**5.73**	16.23	**3.32**
FlyingThings3D	6.64	18.31	5.21
Monkaa	8.47	16.17	7.08
Driving	10.95	**11.09**	9.88

Table 2 FlowNet trained on existing synthetic datasets. Sintel and FlyingChairs were split into a training and a validation set. As expected, training on the Driving dataset works best for KITTI. The Sintel dataset, although very similar to its validation set, is too small to yield great performance. Surprisingly, the FlyingChairs datasets yields consistently better numbers than the more sophisticated FlyingThings3D and Monkaa datasets. This observation motivates our investigation of what makes a good dataset.

These might not be the same properties that make a good test set for benchmarking. Since benchmarking is not the aim of this paper, we use the established benchmark datasets Sintel (Butler et al, 2012) (the "clean" variant) and KITTI 2015 (Menze and Geiger, 2015) to measure the performance of networks trained on different data. While the evaluation on Sintel covers the generalization of the network to other synthetic datasets, the KITTI experiments cover the generalization to real-world data in the restricted setting of a driving scenario. In all cases we report the average endpoint error (EPE).

5.2 What makes a good training dataset?

In Dosovitskiy et al (2015), we showed that the optical flow estimation task does not have to be trained on data which semantically matches the test data: there are no chairs in the Sintel dataset, but the network trained

on the FlyingChairs dataset performs well on Sintel. In fact, Table 2 shows that a network trained on the 22k samples of the training split of the FlyingChairs dataset performs better than a network trained on a subset of the Sintel training dataset with 908 samples and tested on the remaining 133 samples for validation.

This positive result from Dosovitskiy et al (2015) does not yet clearly indicate what are the relevant dataset properties, apart from its size. Interestingly, Table 2 also shows that training on the more diverse and more realistic FlyingThings3D dataset yields inferior performance to training on FlyingChairs. This is surprising and motivates our efforts to get more insights into: (1) which properties of the FlyingChairs dataset make it so successful for training optical flow networks and (2) how the training data can be potentially further improved.

To this end, we performed an extensive ablation study to evaluate the contributions of object shape and types of motion. These aim primarily on explaining the generally good training behavior of the FlyingChairs dataset. Additionally, we tested the importance of surface features imparted by textures and the effect of lighting when lifting the FlyingChairs dataset into 3D, and what happens when combining the FlyingChairs and FlyingThings3D datasets. This is to investigate the underlying reason why the FlyingChairs dataset outperforms the more sophisticated FlyingThings3D. This set of experiments was conducted on the optical flow task.

In a complementary step, focusing on real data, we looked at data characteristics that originate not in the observed scene but in the imaging system itself. In particular, we were interested in how explicit modeling of *e.g.* camera lens distortion or Bayer artifacts in the training data may help improve the performance of the resulting network when applied to images from a real camera system showing these characteristics. For this set of experiments, we used the disparity estimation task because we found comparing disparity maps and assessing fine details by eye much easier than doing so on flow fields.

Due to time and compute power constraints, we could not evaluate both parts of our experiments suite on both optical flow and disparity estimation. Considering how closely related the two tasks are, the disparity results are presumably valid for optical flow as well, just as the optical flow results can likely be applied to disparity estimation.

5.2.1 Object shape and motion

In our first experiment, we investigated the influence of object shapes and the way they move. The general setup here is similar to the FlyingChairs dataset—2D motions of objects in front of a background image—and we used the same composition approach, shown in Fig. 2. However, instead of chairs, we used different randomly shaped polygons and ellipses; and instead of arbitrary affine motions, we used different subsets thereof, plus optionally non-rigid deformations. We used the same random Flickr images as in the FlyingChairs dataset, both for the background and for the foreground objects. On every dataset variant we trained a network from scratch and evaluated its performance on three benchmark datasets: Sintel, KITTI and FlyingChairs.

We designed a series of increasingly complex datasets, starting off with axis-aligned rectangular boxes and motions restricted to translation only, and going all the way to complex thin and non-convex objects with non-rigid motions. Examples for the tested scenarios are shown in Figure 4. During training we applied the same color and geometry augmentations used for optical flow training in Dosovitskiy et al (2015); however, geometry augmentations were restricted to respect the class of motions of the training dataset; for instance, we applied no rotation augmentation if there was no rotation motion in the dataset.

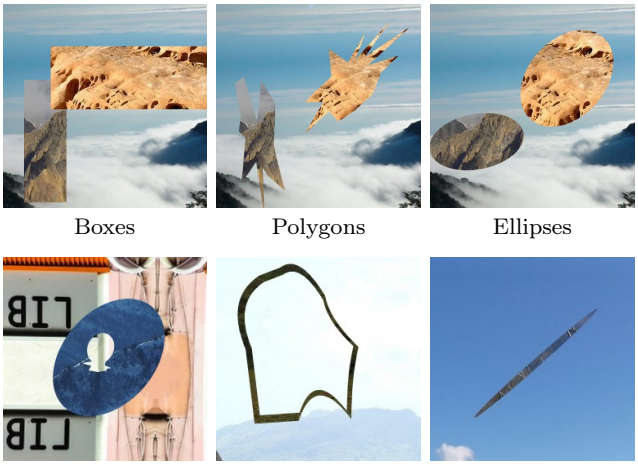


Fig. 4 Object shapes. Distinct shape classes used in our "shapes and motions" ablation experiment. For more examples, see Fig. 11 and Fig. 12.

The results are shown in Table 3. Even the simplest dataset consisting of translated axis-aligned rectangles leads to reasonable performance on the benchmark datasets. As expected, the general trend is that more complexity and diversity in the shapes and motion during training improves the performance of the resulting networks. This can be best seen for the Sintel dataset, where, except for the addition of holes into

Training data	Polygons	Ellipses	Translation	Rotation	Scaling	Holes in objects	Thin objects	Deformations	Sintel train clean	KITTI 2015 train	FlyingChairs
Boxes	(✓)		✓						5.29	17.69	4.95
Polygons	✓		✓						4.93	17.63	4.60
Ellipses		✓	✓						4.88	17.28	4.87
Polygons+Ellipses	✓	✓	✓						4.86	17.90	4.62
Polygons+Ellipses+Rotations	✓	✓	✓	✓					4.79	18.07	4.38
Polygons+Ellipses+Scaling	✓	✓	✓	✓	✓				4.52	15.48	4.22
Polygons+Ellipses+Holes in objects	✓	✓	✓	✓	✓	✓			4.71	16.36	4.20
Polygons+Ellipses+Thin objects	✓	✓	✓	✓	✓	✓	✓		4.60	16.45	4.13
Polygons+Ellipses+Deformations	✓	✓	✓	✓	✓	✓	✓	✓	**4.50**	**14.97**	4.23
FlyingChairs	(✓)	(✓)	✓	✓	✓	✓	✓		4.67	16.23	(**3.32**)

Table 3 Object shape and motion. Each row corresponds to one training set, containing certain object shapes and motion types. For reference, we also show results for a FlowNet trained on the FlyingChairs dataset (Dosovitskiy et al, 2015). The trained networks are tested on three benchmark datasets. Even the simplest training dataset leads to surprisingly good results and adding more complex shapes and motions yields further improvement. Nonrigid deformations help on Sintel and KITTI, but not on the rigid FlyingChairs. On the other hand, the chairs contain holes whereas training with holes in objects weakens the results on Sintel and KITTI. Example images in Fig. 11 and Fig. 12 in the appendix.

objects, each additional level of complexity slightly improves the performance. While adding only rotations does not lead to much better scores and even decreases the score on KITTI, adding scaling motions on top yields a large improvement. Holes in objects seem to be counter-productive, perhaps because objects in benchmark datasets rarely have holes. Nonrigid deformations on objects and background lead to the overall best result.

The observations can be explained by looking at the types of object motion present in each test dataset: adding rotation motion yields the biggest improvement on FlyingChairs which is also the testset with the strongest object rotation. The dominant motion in KITTI is scaling motion induced by the scene moving towards the camera. Thus, training on scaling motion helps in this case. The nonrigid 2D deformations can approximate the flow patterns exhibited by rotating 3D objects; the effect of training with this is noticeable in Sintel, and strongest in KITTI. Our results confirm that training data can be improved if it is possible to reason about the target domain. On the other hand, this is also disappointing because it indicates that there is no single best general-purpose training dataset.

5.2.2 Textures

In the experiments reported in the previous section we used the same real-world photographs obtained from Flickr both as the background and as object textures.

Train data	Test data: Plasma	Clouds	Flickr	Sintel
Plasma	3.57	6.10	5.41	5.85
Clouds	3.74	3.70	5.05	5.08
Flickr	3.88	4.13	4.07	**4.57**

Table 4 Object and background texture. We trained FlowNet with three texture types illustrated in Table 5 and tested on these datasets and on Sintel. Flickr textures (*i.e.* real photographs) yield the best performance on Sintel.

These provide textures with realistic image statistics, but lack natural semantic context. But how much does the choice of textures matter for the performance of the trained networks?

In this section we compare three texture variants, illustrated in Table 5. In addition to Flickr images, we created two additional texture sets: *Plasma* and *Clouds*. "Plasma"-type textures[4] are segmented into cells whose sizes range from few pixels to a quarter of the image width. The cells have clear boundaries; each cell has a single flat color, and cell colors gradually change across the image. The "clouds"-type images were made by composing color noise on multiple scales. The resulting textures exhibit structure on all frequencies, but contain almost no constant-color regions or sharp outlines. These images appear confusing and self-similar to the human eye.

[4] made by ImageMagick's random "plasma" generator

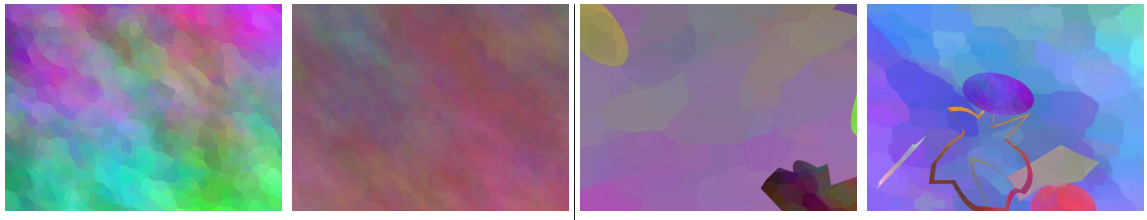
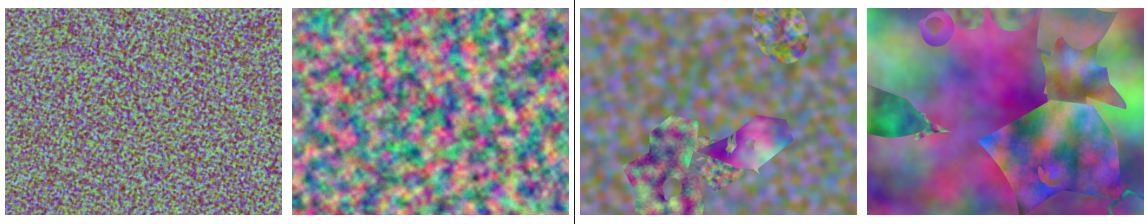
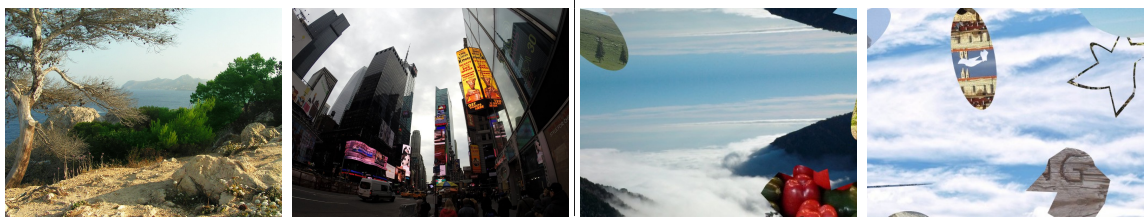
Texture type	Texture samples	Data samples
Plasma		
Clouds		
Flickr		

Table 5 Textures. Example texture images and training samples for the textures ablation experiment. Plasma and Clouds are two types of procedurally generated random textures, while Flickr corresponds to using random natural images from Flickr as textures. More examples in Fig. 12 and Fig. 13.

We trained a network from scratch on each texture set and tested it on each other's testing subsets (a fixed set of 1000 samples each). Table 4 shows that the Flickr textures generalize best to new datasets. In particular, they yield the best results on the Sintel dataset. This shows how important the diversity of textures during training is for the final network performance, and explains the poor performance when training a network on the Monkaa dataset, which comprises complex objects and motion patterns, but highly repetitive and monotonous textures.

Thus, we evaluated how the performance of the network changes if the displacement distribution of the training set deviates from the distribution of the test set. We doubled and tripled the size of the displacements in the FlyingChairs dataset by doubling/tripling all object and background motion distribution parameters (standard deviation for normal distributions, minimum and maximum for uniform distributions). Fig. 5 shows histogram plots of the datasets' displacement magnitudes; "2x/3x Sintel-like" are the datasets with exaggerated displacements.

5.2.3 Displacement statistics

Classical optical flow estimation methods are sensitive to the displacement magnitude. Variational methods (Horn and Schunck, 1981) have clear advantages in case of small displacements, whereas large displacements require extra treatment and the use of additional combinatorial techniques (Brox et al, 2004; Brox and Malik, 2011). When approaching the problem with a deep network, it is not obvious whether similar restrictions apply. However, one would expect that the distribution of displacements in the training data plays a crucial role. This section picks up and expands on experiments done in Ilg et al (2017).

The results are shown in Table 6. Clearly, matching the displacement distribution of the test set is important. Using the same displacement statistics as Sintel (denoted as "Sintel-like") leads to the best results, whereas large deviations from these statistics lead to a large drop in performance. In general, such importance of matching the displacement statistics of the test data is disappointing, since one would like to have a network that can deal with all sorts of displacements. This problem can be largely alleviated by the use of learning schedules, as described in Section 5.3, and by architectures that mix specialized networks, as proposed by Ilg et al (2017).

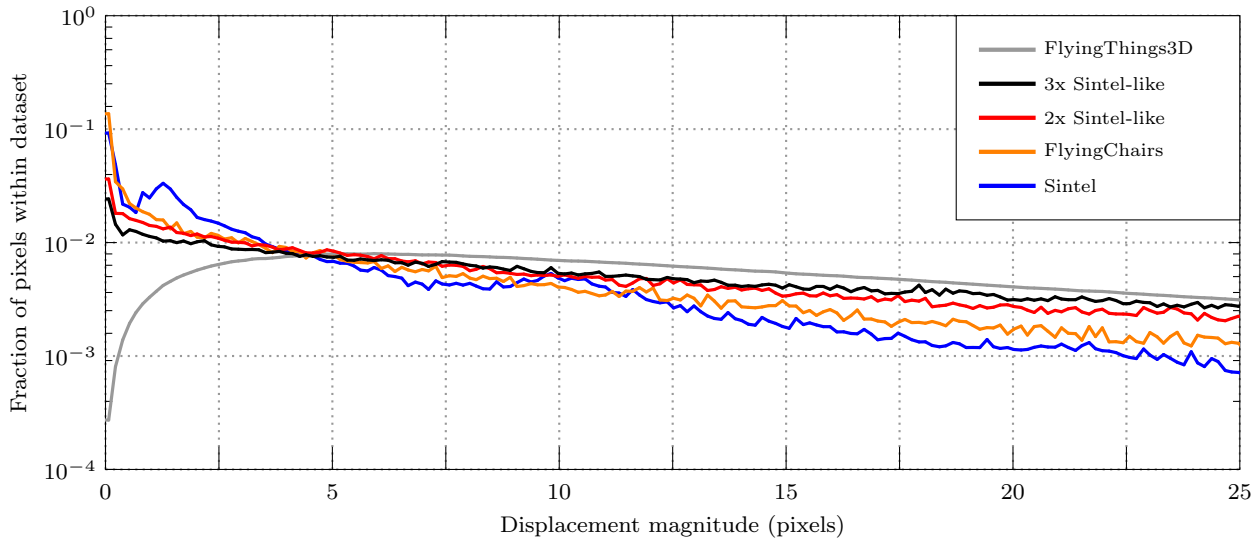


Fig. 5 Displacement statistics. The statistics of the displacement size of the FlyingChairs dataset matches the statistics of the Sintel dataset fairly closely. The FlyingThings3D dataset was not tuned to match any specific histogram and contains relatively few small displacements. Its greater total count of pixels makes its histogram curve appear smoother. The "2x/3x Sintel-like" datasets intentionally exaggerate Sintel's distribution (see Section 5.2.3).

Train data	Test data			
	Sintel-like	2x Sintel-like	3x Sintel-like	Sintel
Sintel-like	4.17	11.25	21.84	**4.60**
2x Sintel-like	5.04	11.68	20.82	4.99
3x Sintel-like	5.91	12.73	21.70	5.75

Table 6 Flow magnitude distributions. Results when training on data with different motion statistics. The more the histogram differs from Sintel's, the worse a trained network performs on Sintel. "2x/3x Sintel-like" are the same as in Fig. 5.

Displacement Range	Partial EPE		
	Trained on FlyingChairs	Trained on FlyingThings3D	Difference
0–∞px	**4.67**	5.76	1.09
0–10px	**0.87**	1.46	0.49
10–40px	**1.22**	1.56	0.34
40–160px	**2.07**	2.33	0.26
160–∞px	0.51	**0.42**	−0.09

Table 7 Performance depending on displacement magnitude. Comparison of error contributions when testing FlowNet on Sintel train clean. Training on FlyingThings3D improves only for displacements larger than 160 pixels but is in all other cases worse. Thus, the displacement statistics from Fig. 5 cannot be the only reason for inferior performance when training on FlyingThings3D.

5.2.4 Lighting

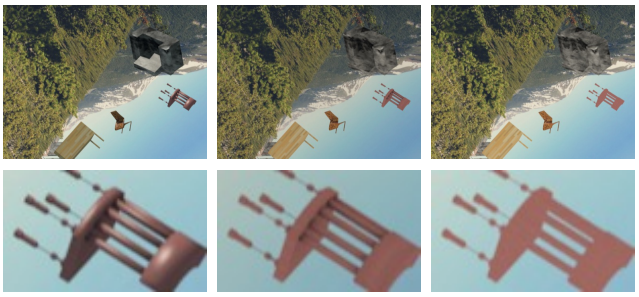
In the real world (and in realistically lit synthetic data, such as Sintel), the interaction between light and object surfaces generates reflections, highlights, and shadows. These phenomena pose a challenge to vision algorithms because they can violate the photoconsistency assumption. Indeed, highlights and shadows can seem like moving "objects", but it is generally desired that these effects be ignored.

Potentially, deep networks can learn to distinguish lighting effects from texture and ignore the false motion due to lighting. To this end, they must be provided with training data that shows correct lighting effects. We tested whether the FlowNet is able to exploit sophisticated lighting models during training or if optical flow can be estimated just as well with simpler, potentially more efficiently computed lighting.

For this experiment, we used the FlyingChairs dataset re-modeled in 3D, as described in Section 3.1.2, in order to apply lighting effects. We rendered the three different lighting settings shown in Fig. 6: shadeless, static, and dynamic. More examples are shown in Fig. 14 in the appendix.

The objects in the shadeless variant show no lighting effects at all. Some object models are textured, but many simply have a uniform flat color. Estimating optical flow for these objects is hard because the only usable features are given by the shape of the object contour.

Statically lit objects are shaded by environmental lighting which is uniform in all directions and does not



dynamic static shadeless

Fig. 6 Lighting Quality. Data samples and crops from our "lighting realism" ablation experiment. We used three different render settings, each on the same raw data. The background images are always flat photographs. The crops in the lower row appear blurred only due to upsampling for visualization. See Fig. 14 for more examples.

	Test data			
Train data	shadeless	static	dynamic	Sintel
shadeless	2.71	2.77	2.93	4.41
static	2.69	2.67	2.79	**4.16**
dynamic	2.94	2.85	2.74	4.22
mixture	2.83	2.80	2.86	4.24

Table 8 Lighting. Comparison of the lighting models shown in Fig. 6. A network trained on realistic dynamic lighting also expects realistic lighting in the test data. A network trained on static lighting cannot exploit dynamic lighting features, but performs best on Sintel whose realistic lighting setup generates both "static" and "dynamic" effects. A 1:1 static/dynamic mixture does not perform better.

consider self-shadowing or other objects. This corresponds to applying a fixed "shadow texture" to each object. Known as *ambient occlusion baking*, this approach is often used in computer graphics to make objects look more realistic without having to recompute all lighting whenever an object or light moves.

The dynamic lighting scenario uses raytracing and a light source (lamp), shining into the scene from a randomized direction. Objects in this case cast realistic shadows and if an object rotates, the shading on its surface changes accordingly. The chairs also show specular highlights. This scenario presents the most realistic effects. It allows the network to learn about lighting effects, but this also makes the learning task harder: the network must learn to distinguish between different materials and that in case of specular materials, the changing position of a highlight is not supposed to induce optical flow.

We trained a separate network from scratch on each scenario and tested them on each other's test data, as well as on Sintel's clean training data. The results in Table 8 show that the network can exploit more complex lighting in the training data to perform better on test data. The largest effect of lighting, though, is the larger number of visual features on object surfaces. The effect of dynamic lighting is much smaller because highlights are sparse in the image.

The results also show that the network trained with dynamic lighting gets confused by objects that have not been generated with dynamic lighting. This can also explain why this network performs marginally worse on Sintel than the network trained with static lighting: if there is no strong directional light source in a Sintel scene, the objects have "static" features but not the hard shadows produced in our "dynamic" setup. Some surfaces in Sintel and in real scenes are Lambertian, others shiny. The network must learn to distinguish the different surface materials to fully exploit the lighting cue. A training set that only contains chairs is possibly not sufficient to capture different surface materials. Moreover, this makes the learning task more difficult. The latter would also partially explain why the FlyingThings3D dataset, which comes with realistic lighting and many object categories, leads to a network that is inferior to one trained on FlyingChairs. This interpretation is supported by the experiments in Section 5.3.

Color changes		Flow changes		
on both frames	between frames	on both frames	between frames	Sintel
				7.66
✓				6.30
✓	✓			6.12
		✓		5.25
		✓	✓	5.33
✓		✓		5.11
✓	✓	✓	✓	**4.60**

Table 9 Data augmentation. The results show the effect on FlowNet performance when different types of augmentation are applied to training data (here we used "+Thin objects" from Table 3). The data can be augmented by applying color and/or geometry changes. One can also choose (a) to only apply the same changes to both input images, or (b) to additionally apply a second set of incremental changes to the second image.

5.2.5 Data augmentation

We split the data augmentation into augmentations on color and augmentations on geometry (these augmentations were used in our previous works (Dosovitskiy et al, 2015; Mayer et al, 2016; Ilg et al, 2017), but there we did not analyze the effects of the individual options). In Table 9 we show an ablation study on the full set of augmentation options: no augmentation, augmentations on either color or geometry, or both. We further split the

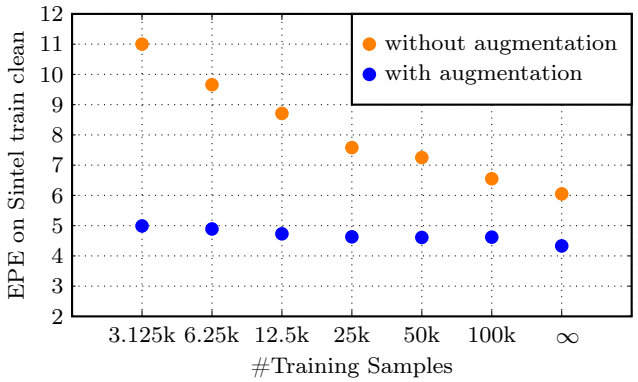


Fig. 7 Amount of Training Data. ∞ indicates as many training samples as training iterations (no training sample was ever used twice). This figure complements the augmentation modes experiment in Table 9 and uses the same training data setting. A FlowNet trained without data augmentation gets much greater relative benefits from more training data, compared to augmented training. However, even with infinite data, networks without augmentation perform far worse.

experiments by whether both images of a sample pair receive only the same augmentation, or whether a second, incremental transformation is additionally applied to the second image.

The evaluation shows that almost all types of data augmentation are complementary and important to improve the performance of the network. The notable exception is additional geometry augmentation between the frames of a sample. The raw dataset seems to already contain sufficiently varied displacements, such that additional augmentation between frames cannot generate helpful new data. Contrary to this, color changes between frames always introduce new data: the unaugmented dataset does not contain this effect, and applying the same color augmentation to both frames does not change this.

5.2.6 Amount of training data

It is generally accepted that more training data leads to better results in supervised training. Data augmentation increases the effective size of the training dataset. However, an augmented existing sample is not the same as a truly new sample: for example, augmentation as we use it cannot create new textures or change the relative positions within a group of objects.

We compared the effect of augmentation to truly changing the size of the dataset. To evaluate this, we trained FlowNets on different amounts of data and either allowed full data augmentation or none (corresponding to the first and last rows in Table 9). The results are given in Fig. 7 and show that while more data is indeed always better, randomized augmentation yields another huge improvement even for an infinite amount of training samples, *i.e.* as many samples as training iterations (corresponding to 2.4M samples for 600k minibatches and batch-size 4). Therefore, augmentation allows for a $\sim$100-fold reduction of the training data and still provides better results.

From this section and the previous Sec. 5.2.5, we conclude that data augmentation serves two purposes: (a) if it only replicates the effects found in the original data, then augmentation increases the effective size of the training set without changing the domain of the data. This can help if the dataset is just not big enough. On the other hand, (b) augmentation may be able to cover types of data variation which are complementary to that of the original data (depending on the task and data domain), and in this case a network can learn to cope with a wider range of inputs. However, augmentation has inherent limits, and we achieved the best results when using both augmentation and as much raw data as possible.

5.3 Learning schedules with multiple datasets

The results from Table 2 in Section 5.2 show that training a network on the FlyingThings3D dataset (Ilg et al, 2017) yields worse results on Sintel than training the same network on FlyingChairs (Dosovitskiy et al, 2015), although the former is more diverse and uses more realistic modeling. The experimental results presented so far do not yet give an explanation for this behavior. This section complements the dataset schedule experiments from Ilg et al (2017).

Fig. 5 reveals that FlyingThings3D has much fewer small displacements (smaller than 5 pixels). Moreover, Table 7 shows that the displacement statistics cannot be the only reason for the inferior performance of FlyingThings3D over FlyingChairs: although displacements between 40 and 160 pixels are well represented in FlyingThings3D, it still performs worse than FlyingChairs in that displacement range.

To investigate the effects of both datasets further, we trained FlowNets on combinations of both datasets, using the S_{long} and S_{fine} schedules (see Fig. 8). The results are given in Table 10 and show that, in fact, using a combination of FlyingChairs and FlyingThings3D training data yields the best results. Moreover, this only holds if the datasets are used separately and in the specific order of training on FlyingChairs first and then fine-tuning on FlyingThings3D. This reveals that not only the content of the data matters, but also the point of time at which it is presented to the network during the training phase. This is strongly connected to curriculum learning, which has previously been applied

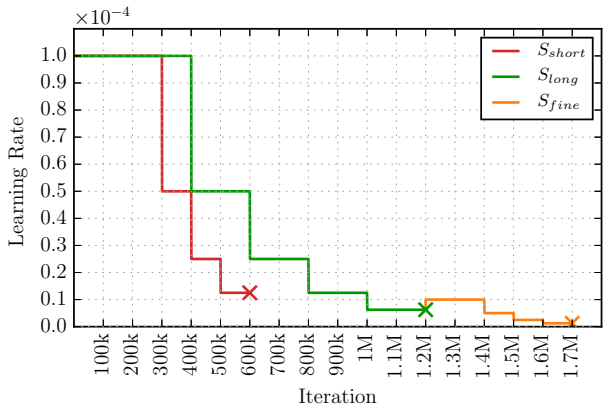


Fig. 8 Learning rate schedules. In most experiments of this work S_{short} is used due to its lower computational cost. Section 5.3 uses S_{long} in combination with S_{fine}, where S_{fine} is used for fine-tuning on a second dataset (as done in Ilg et al (2017) where these schedules were introduced).

Datasets	S_{long}	S_{fine}
FlyingChairs	4.22	4.15
FlyingThings3D	5.14	4.93
Mixture	4.31	4.25
FlyingChairs → FlyingThings3D	4.22	**4.05**
FlyingThings3D → FlyingChairs	5.14	4.60

Table 10 Learning schedules. Results on Sintel train clean of training FlowNet on mixtures of FlyingChairs and FlyingThings3D with schedules S_{long} and S_{fine}. "Mixture" is a 1:1 mix of both entire datasets. One can observe that only the sequential combination FlyingChairs → FlyingThings3D gives significantly better results.

to shape recognition (Bengio et al, 2009; Elman, 1993) and other (non-vision) tasks. From this observation, we conclude that presenting a too sophisticated dataset too early is disadvantageous, but at a later stage it actually helps. A reason might be that the simpler FlyingChairs dataset helps the network learn the general concept of finding correspondences without developing possibly confusing priors for 3D motion too early.

5.4 Synthesizing the defects of real imaging

In this section, we focus on the artifacts that real-world imaging systems add to the data. The effects of the imaging process on the obtained images have recently been studied by Klein and Murray (2010). There, a rendering pipeline was designed to artificially apply the same artifacts that the real camera produces to rendered objects. As a result, augmented-reality objects blended well into the real images; without this adjustment, the rendered objects stood out because they looked unnaturally clean and clear. The OVVV dataset by Taylor et al (2007) uses a comparable approach, simulating analog video effects to make their evaluation data for video surveillance systems more realistic.

We took this idea and applied it to *training* data: we usually train networks on synthetic data because real data with good ground truth is not always available in quantity or even possible; yet the fact that this synthetic data was not recorded by a physical camera already makes it different. We investigated what happens when we corrupt the training data with the same flaws of real cameras[5].

Applying such data degradation leads to two possible, counteracting effects: (a) in artificially degraded data, the relationship between image pixels and ground truth labels is muddied by blur etc. The network already has to cope with the inherent discretization artifacts of raster images in any case, but complicating this further could lead to worse performance. On the other hand, (b) the network should be able to learn that certain flaws or artifacts convey no meaning and subsequently be able to ignore such effects. If the network can transfer this knowledge to real-world test data, its performance should improve.

We performed two disparity estimation experiments to test our conjectures: we observed the qualitative effects of degraded synthetic training data on images taken with a commercial stereo camera with low-quality wide-angle lenses; and we quantified whether training data adapted to camera artifacts in KITTI can improve DispNet performance on the KITTI2015 dataset (one of the few real-world datasets with ground truth). In both cases, the degradation was tuned to simulate the actual lens and camera effects seen in the respective real data.

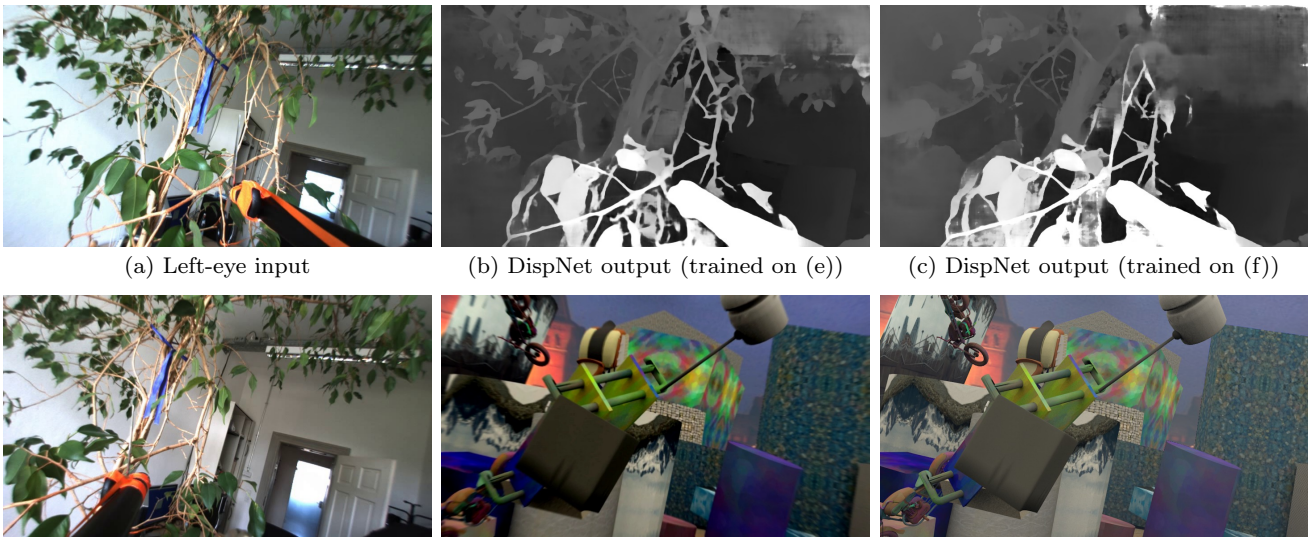
Note that this section's view on data is orthogonal to that of the preceding sections, and we consider the experiments disjoint: above, we looked at *features and content* in the data (motion types, texture patterns, object shapes etc.). In this section, we rather look at *pixel-level* data characteristics that are independent from what the images actually depict. To perform these experiments, we used the established datasets from our prior works, not the additional experimental ones from this paper.

5.4.1 Lens distortion and blur

The notable degradation of images produced by a Bumblebee[6] stereo camera is due to radial blur (caused

[5] Visual effects artists have long been using the knowledge that "perfect" pictures are not perceived as "real" by humans; hence artificial film grain, chromatic aberrations, and lens flare effects are applied in movies and computer games.

[6] Bumblebee2 BB2-08S2C



(a) Left-eye input (b) DispNet output (trained on (e)) (c) DispNet output (trained on (f))

(d) Right-eye input (e) Degraded training data (f) Original training data

Fig. 9 Lens distortion and blur in training data. A DispNet trained on clean synthetic data (f) does not perform well out of the box on images from a real camera; the input images from a Bumblebee stereo camera (a,d) show significant radial blur after undistortion, as well as general blur. Adding these degradation effects to the training data (e) to make it look more like the images from this camera leads to significantly more detailed disparity estimates (b) than without these effects (c), especially far away from the image center, where the radial blur is the strongest. In this case, the network learns to deal with the faults in the training data and can transfer this knowledge to the real images.

by undistortion from a wide-angle lens), some general blur, and over-saturated colors (*i.e.* strong contrast). We replicated these effects offline for the entire FlyingThings3D training dataset (Mayer et al, 2016), using filters in the GIMP image processing tool.

To test how the degradation affects performance, we trained two DispNetCorr1D networks from scratch (Mayer et al, 2016): one on the original dataset and the other on the degraded data. Fig. 9 highlights the qualitative differences in the disparity maps estimated by the two networks: training on artificially degraded data produces much finer details, especially in areas towards the image boundaries, where the lens distortion has the largest effect. The network even makes more sensible guesses for areas which are occluded in the other view: the network seems to have learned that object boundaries can be blurry. This cannot be inferred from the original training data with crisp contours.

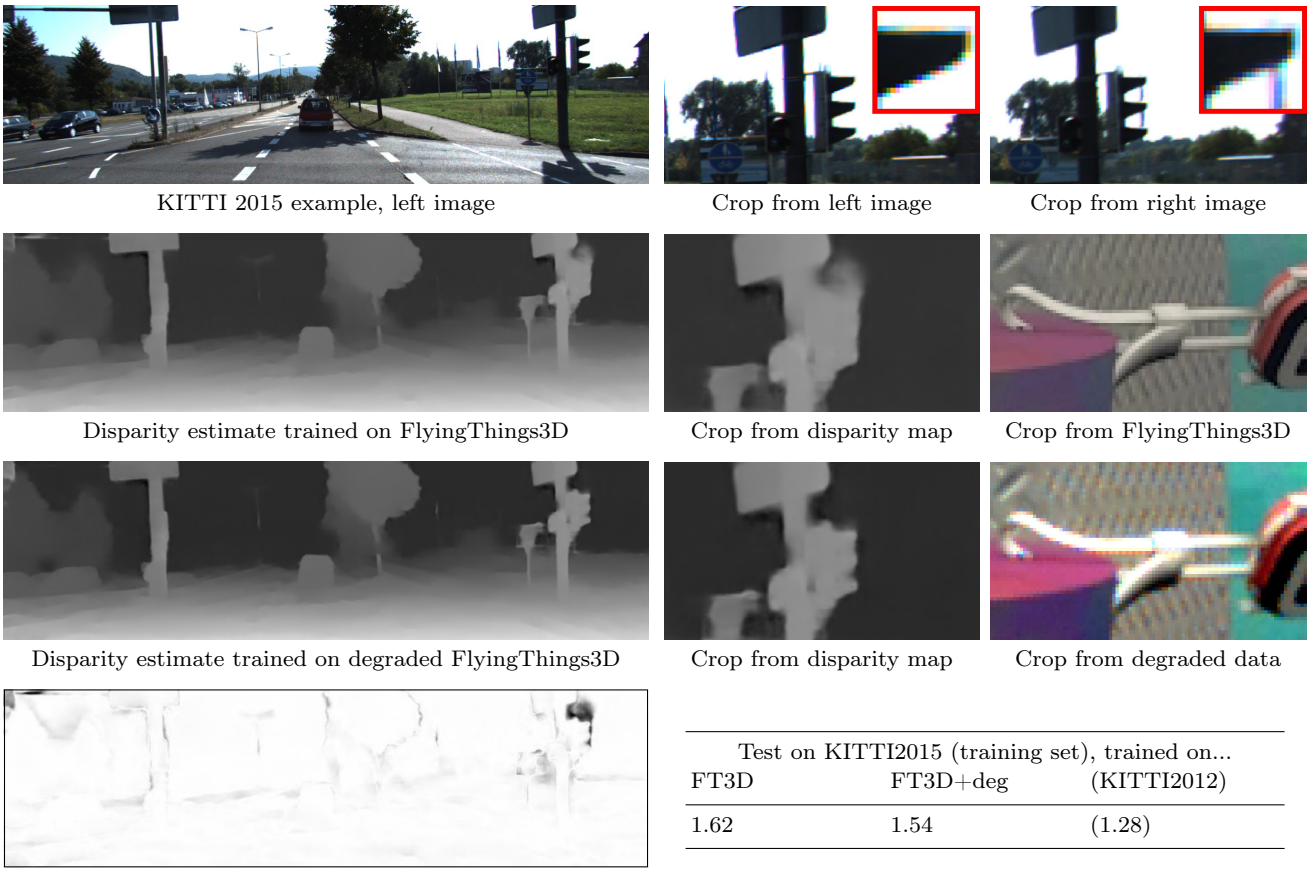
5.4.2 Bayer-interpolation artifacts

Imaging sensors can generally sense only light intensity, not color. To get color images from such a sensor, the most common strategy is to cover the sensor with a repeating pattern of color filters, such that each pixel becomes sensitive to a certain part of the color spectrum. The classic way to do this is the Bayer pattern. To reconstruct a full-color image, each pixel's color channels must be inferred from within a neighborhood, which can lead to significant artifacts. To test whether a network can be preconditioned to deal with Bayer-interpolation artifacts, we emulated them on the FlyingThings3D dataset by first simulating how a Bayer sensor would perceive the scene and then interpolating an RGB image from this virtual sensor image. We evaluated the effects by finetuning a DispNet on FlyingThings3D data with or without Bayer artifacts and then testing the network on the KITTI 2015 training set.

Fig. 10 shows examples of our original and degraded data, as well as the effects on the network output. Our degraded data improves the KITTI 2015 score by 5%. This shows that adding these effects helps, but from the qualitative results one might expect a larger difference. This is because the KITTI ground truth is sparse and has gaps along object contours, due to the nature of its acquisition method using a laser scanner and the perspective difference between the scanner and the camera. A difference image between the disparity estimates of our two synthetic-data networks reveals that most changes actually occur at object boundaries, and the highlighted crops are located in an image region for which there is no ground truth at all.

6 Conclusion

In this paper, we performed a detailed analysis of the synthetic datasets for deep network training that we



Test on KITTI2015 (training set), trained on...		
FT3D	FT3D+deg	(KITTI2012)
1.62	1.54	(1.28)

Fig. 10 Bayer-interpolation artifacts in training data. Color fringing (first row) from naive Bayer-pattern interpolation poses problems to a DispNet trained+finetuned on clean synthetic FlyingThings3D data (second row). Finetuning on a purposefully degraded version of the same dataset instead helps the network deal with this effect (third row). The disparity difference image highlights that the two networks' outputs differ mostly along object contours where the artifacts have the greatest impact. The accompanying table shows that degraded training data produces better results than the artifact-free original data. The (rarely possible) best-case scenario of training with ground truth data from the target domain still yields the best results; this is expected as this experiment only targets low-level data characteristics, and FlyingThings3D data (degraded or not) cannot teach disparity priors for KITTI's street scenes.

introduced in earlier conference papers for optical flow and disparity estimation. This analysis led to several findings, of which we summarize the most important ones here: (1) Diversity is important. A network trained on specialized data generalizes worse to other datasets than a network trained on diverse data. (2) Realism is overrated. Most of the learning task can be accomplished via simplistic data and data augmentation. Realistic effects, such as sophisticated lighting models, are not important to learn basic optical flow, but merely induce minor improvements. (3) Learning schedules matter. Learning schedules that combine multiple different datasets, especially simpler and more complex ones, are a way to greatly improve the generic performance of the trained networks. (4) Camera knowledge helps. Modeling distortions of the camera in the training data largely improves the network's performance when run with this camera.

According to our evaluation, these findings are valid for optical flow (1–3) and disparity estimation (4). They may not hold for high-level tasks such as object recognition. With an increasing necessity to jointly solve low-level and high-level tasks, we see a major future challenge in designing datasets that can serve both regimes and in finding learning procedures that can efficiently combine the advantages of real-world and synthetic data.

References

Aubry M, Maturana D, Efros A, Russell B, Sivic J (2014) Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In:

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 5

Baker S, Scharstein D, Lewis JP, Roth S, Black MJ, Szeliski R (2011) A database and evaluation methodology for optical flow. International Journal of Computer Vision (IJCV) 2, 3, 4

Barron JL, Fleet DJ, Beauchemin SS (1994) Performance of optical flow techniques. International Journal of Computer Vision (IJCV) 2, 4

Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, pp 41–48 15

Brox T, Malik J (2011) Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 11

Brox T, Bruhn A, Papenberg N, Weickert J (2004) High accuracy optical flow estimation based on a theory for warping. Computer Vision-ECCV 2004 pp 25–36 11

Butler DJ, Wulff J, Stanley GB, Black MJ (2012) A naturalistic open source movie for optical flow evaluation. In: European Conference on Computer Vision (ECCV) 2, 3, 4, 8

Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, Xiao J, Yi L, Yu F (2015) ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. ArXiv preprint arXiv:1512.03012 3, 5

Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1

Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1

Dosovitskiy A, Fischer P, Ilg E, Häusser P, Hazırbaş C, Golkov V, van der Smagt P, Cremers D, Brox T (2015) FlowNet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV) 2, 3, 4, 7, 8, 9, 10, 13, 14, 22

Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) Carla: An open urban driving simulator. In: Conference on Robot Learning, pp 1–16 3, 5

Dwibedi D, Misra I, Hebert M (2017) Cut, paste and learn: Surprisingly easy synthesis for instance detection. In: The IEEE International Conference on Computer Vision (ICCV) 2

Eigen D, Puhrsch C, Fergus R (2014) Depth map prediction from a single image using a multi-scale deep network. In: Conference on Neural Information Processing Systems (NIPS) 1

Elman J (1993) Learning and development in neural networks: The importance of starting small. Cognition 48(1):71–99 15

Gaidon A, Wang Q, Cabon Y, Vig E (2016) Virtual worlds as proxy for multi-object tracking analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3, 4, 5

Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? the kitti vision benchmark suite. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2, 3, 6

Handa A, Whelan T, McDonald J, Davison A (2014) A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In: IEEE International Conference on Robotics and Automation (ICRA) 3

Handa A, Pătrăucean V, Badrinarayanan V, Stent S, Cipolla R (2016) Understanding realworld indoor scenes with synthetic data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3

Heeger DJ (1987) Model for the extraction of image flow. Journal of the Optical Society of America (JOSA A) 2

Horn BKP, Schunck BG (1981) Determining optical flow. Artificial Intelligence 11

Huguet F, Devernay F (2007) A variational method for scene flow estimation from stereo sequences. In: IEEE International Conference on Computer Vision (ICCV) 4

Ilg E, Mayer N, Saikia T, Keuper M, Dosovitskiy A, Brox T (2017) Flownet 2.0: Evolution of optical flow estimation with deep networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3, 7, 8, 11, 13, 14, 15

Klein G, Murray DW (2010) Simulating low-cost cameras for augmented reality compositing. IEEE Transactions on Visualization and Computer Graphics 15

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: European Conference on Computer Vision (ECCV) 1

Mayer N, Ilg E, Häusser P, Fischer P, Cremers D, Dosovitskiy A, Brox T (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE Conference on Com-

puter Vision and Pattern Recognition (CVPR) 2, 3, 4, 5, 6, 8, 13, 16, 22

McCane B, Novins K, Crannitch D, Galvin B (2001) On benchmarking optical flow. Computer Vision and Image Understanding 2, 4, 5

McCormac J, Handa A, Leutenegger S, Davison AJ (2017) Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In: The IEEE International Conference on Computer Vision (ICCV) 3

Meister S, Kondermann D (2011) Real versus realistically rendered scenes for optical flow evaluation. In: ITG Conference on Electronic Media Technology (CEMT) 2

Menze M, Geiger A (2015) Object scene flow for autonomous vehicles. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2, 3, 4, 8

Movshovitz-Attias Y, Kanade T, Sheikh Y (2016) How useful is photo-realistic rendering for visual learning? In: ECCV Workshops 3

Onkarappa N, Sappa AD (2014) Speed and texture: An empirical study on optical-flow accuracy in ADAS scenarios. IEEE Transactions on Intelligent Transportation Systems 2

Otte M, Nagel HH (1995) Estimation of optical flow based on higher-order spatiotemporal derivatives in interlaced and non-interlaced image sequences. Artificial Intelligence 2, 4

Qiu W, Yuille AL (2016) Unrealcv: Connecting computer vision to unreal engine. In: Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III, pp 909–916 3

Richter SR, Vineet V, Roth S, Koltun V (2016) Playing for data: Ground truth from computer games. In: European Conference on Computer Vision (ECCV) 3, 5

Richter SR, Hayder Z, Koltun V (2017) Playing for benchmarks. In: International Conference on Computer Vision (ICCV) 5

Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3234–3243 3

Scharstein D, Hirschmüller H, Kitajima Y, Krathwohl G, Nešić N, Wang X, Westling P (2014) High-resolution stereo datasets with subpixel-accurate ground truth. In: Pattern Recognition 3

Silberman N, Hoiem D, Kohli P, Fergus R (2012) Indoor segmentation and support inference from rgbd images. In: European Conference on Computer Vision (ECCV) 1, 3

Song S, Yu F, Zeng A, Chang AX, Savva M, Funkhouser T (2017) Semantic scene completion from a single depth image. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3

de Souza CR, Gaidon A, Cabon Y, Peña AML (2017) Procedural generation of videos to train deep action recognition networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp 2594–2604 3

Sturm J, Engelhard N, Endres F, Burgard W, Cremers D (2012) A benchmark for the evaluation of rgb-d slam systems. In: International Conference on Intelligent Robot Systems (IROS) 3

Su H, Qi CR, Li Y, Guibas LJ (2015) Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views. In: IEEE International Conference on Computer Vision (ICCV) 3

Taylor GR, Chosak AJ, Brewer PC (2007) Ovvv: Using virtual worlds to design and evaluate surveillance systems. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3, 15

Vaudrey T, Rabe C, Klette R, Milburn J (2008) Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In: International Conference on Image and Vision Computing 2

Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3d shapenets: A deep representation for volumetric shapes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3

Wulff J, Butler DJ, Stanley GB, Black MJ (2012) Lessons and insights from creating a synthetic optical flow benchmark. In: ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation 2, 6

Xiao J, Owens A, Torralba A (2013) Sun3d: A database of big spaces reconstructed using sfm and object labels. In: IEEE International Conference on Computer Vision (ICCV) 3

Zhang Y, Qiu W, Chen Q, Hu X, Yuille AL (2016) Unrealstereo: A synthetic dataset for analyzing stereo vision. Tech. Rep. ArXiv preprint arXiv:1612.04647 3

Zhang Y, Song S, Yumer E, Savva M, Lee JY, Jin H, Funkhouser T (2017) Physically-based rendering for indoor scene understanding using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 3

Fig. 11 Dataset gallery 1/4: Boxes, Polygons and Ellipses variants as used in Sec. 5.2.1 (see also Table 3).

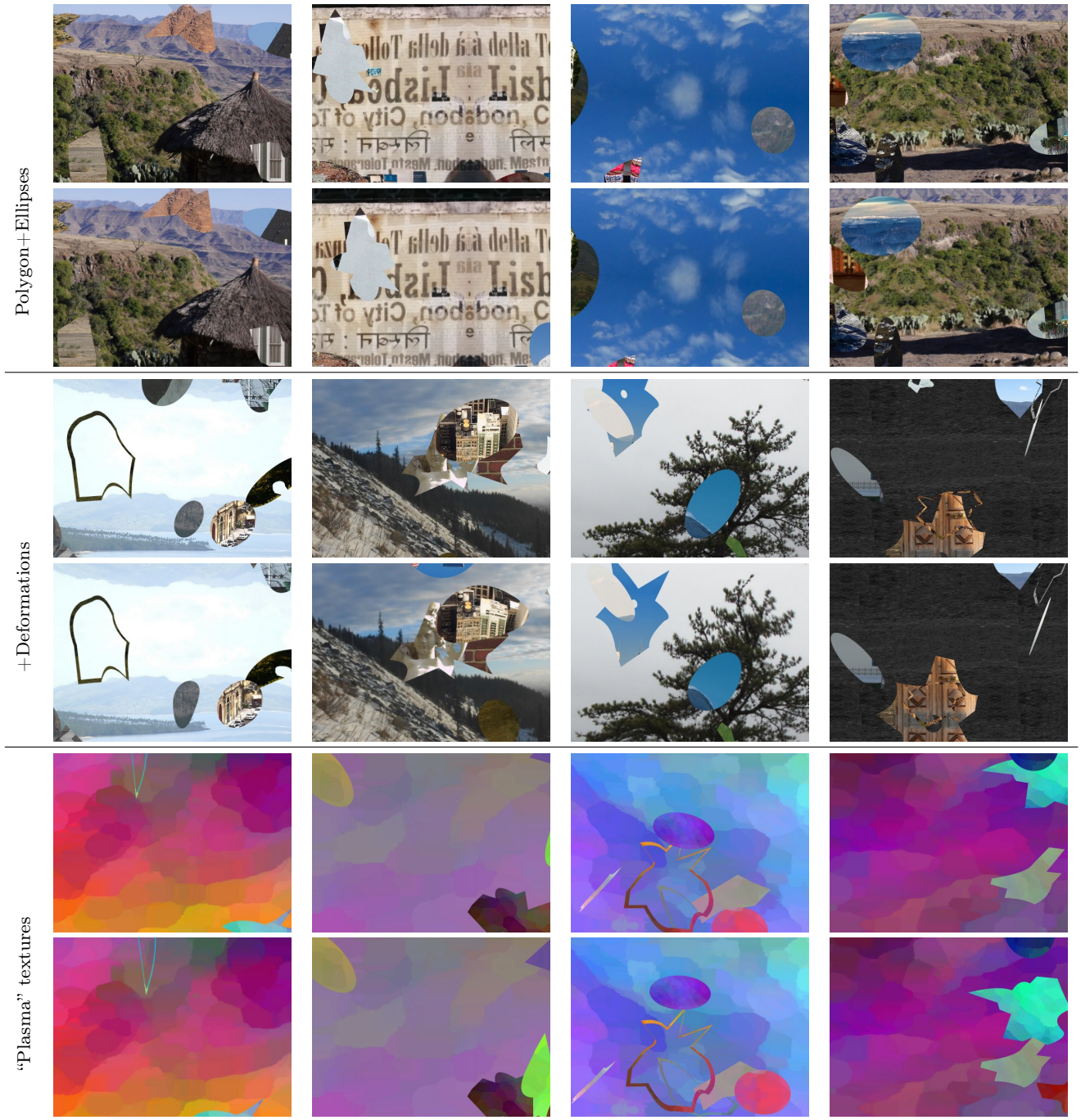


Fig. 12 Dataset gallery 2/4: Polygons+Ellipses and Polygons+Ellipses+Deformations variants as used in Sec. 5.2.1 (see also Table 3); "Plasma" textures as used in Sec. 5.2.2.

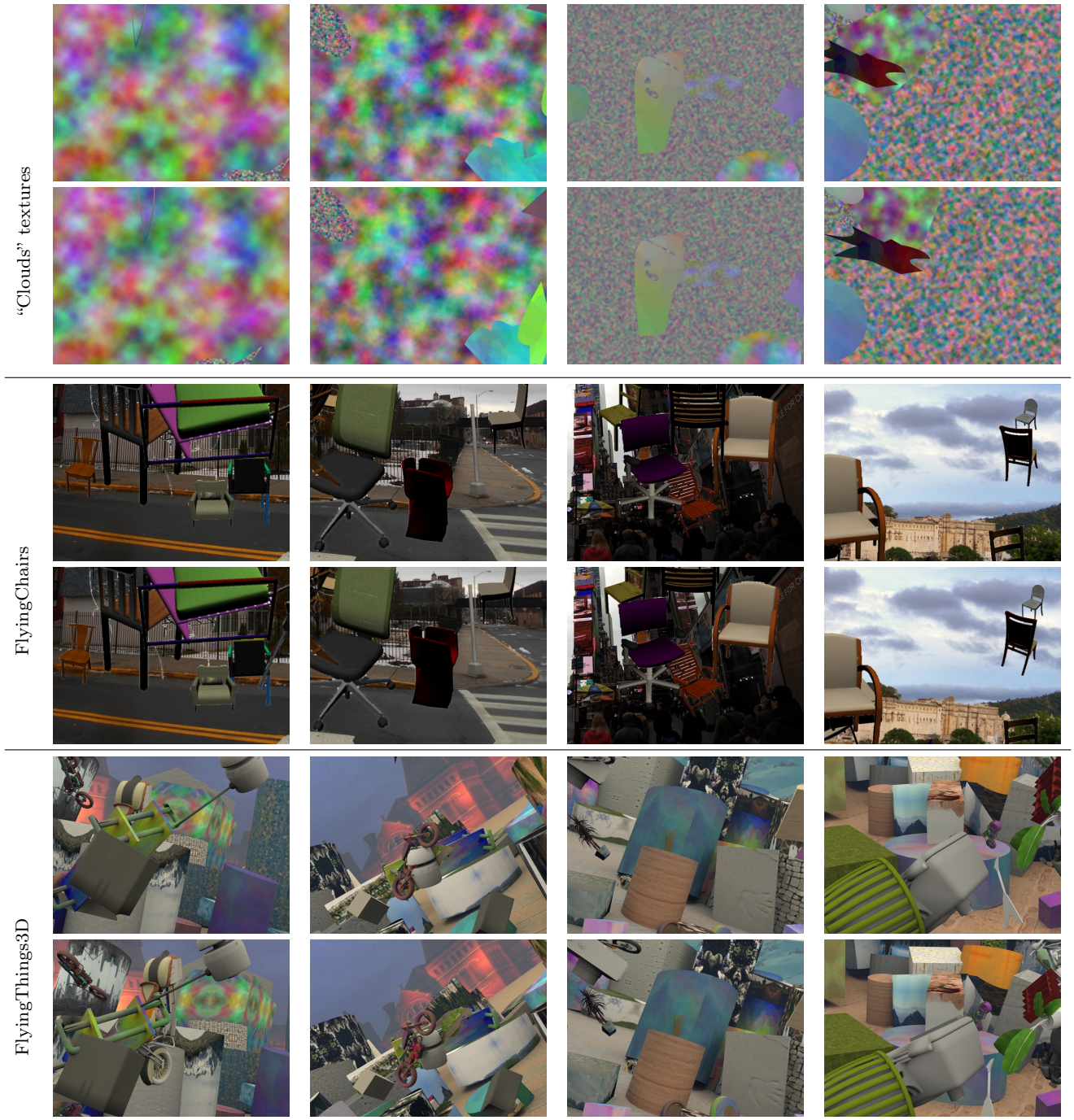


Fig. 13 Dataset gallery 3/4: "Clouds" textures as used in Sec. 5.2.2; FlyingChairs from Dosovitskiy et al (2015); FlyingThings3D (cropped to the same 4:3 aspect ratio as the other datasets for display purposes) from Mayer et al (2016).

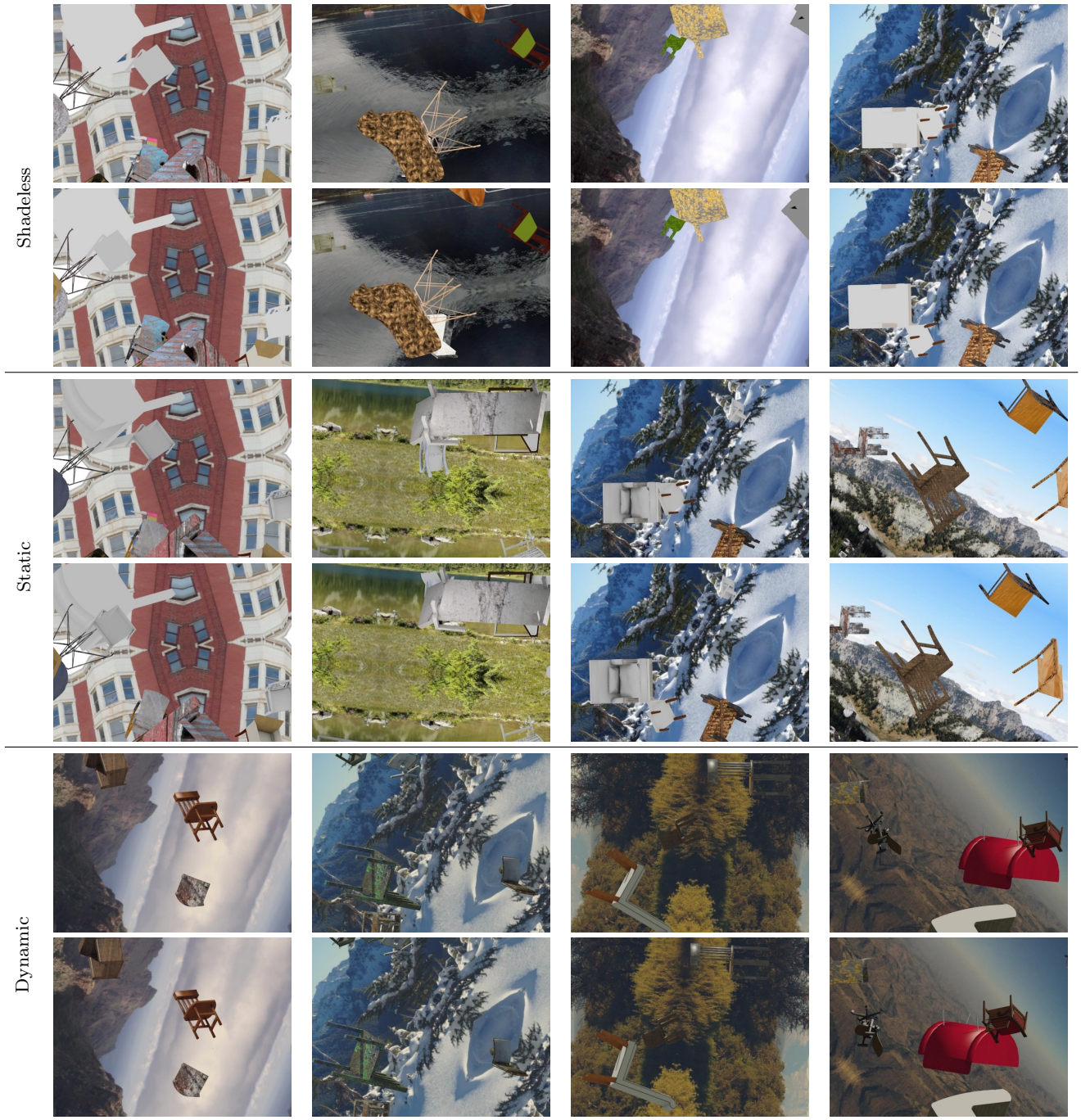


Fig. 14 Dataset gallery 4/4: Shadeless, static and dynamic lighting variants as used in Sec. 5.2.4.