Video Frame Interpolation via Adaptive Convolution

Simon Niklaus, Long Mai and Feng Liu

Portland State University

CVPR 2017

UNIVERSITY OF ALBERTA

Motivation

- Frame rate conversion
- Slow motion video
- Novel view synthesis
- Image morphing

Summary

- Learn a deep CNN to produce a convolutional kernel for each pixel
- Convolve kernel with pixel centered patches from input images to generate corresponding interpolated pixel
- Train directly from unlabeled videos

- CVPR 2017
- ICCV 2017

S. Niklaus, L. Mai and F. Liu, "Video Frame Interpolation via Adaptive Separable Convolution", ICCV 2017 2

Existing Methods – Matching based

- Two step approach:
  - Dense motion estimation
    - Stereo matching
    - Optical flow
  - Pixel synthesis
- Cannot handle challenging scenarios:
  - occlusions
  - out of focus and motion blur
  - sudden illumination changes
  - lack of texture

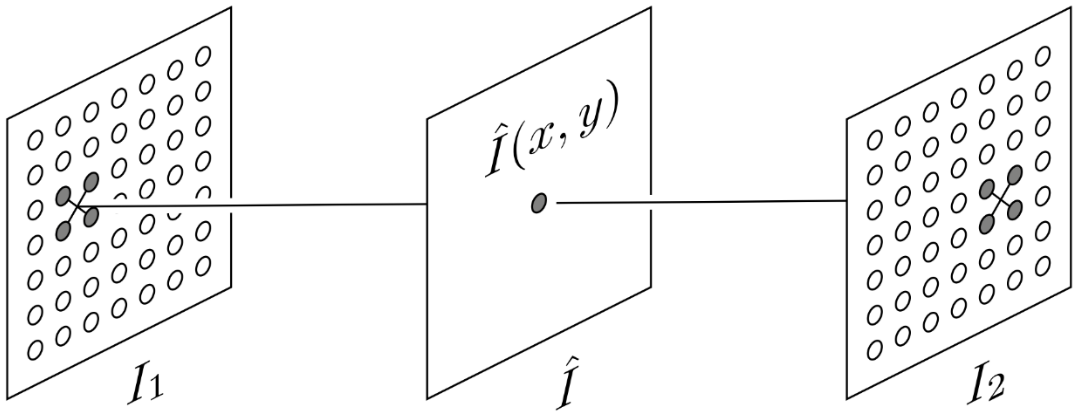
Existing Methods – Phase based

- Fourier domain methods
- Motion encoded by phase shift
- Cannot estimate fast motion
- Do not handle high frequency components well

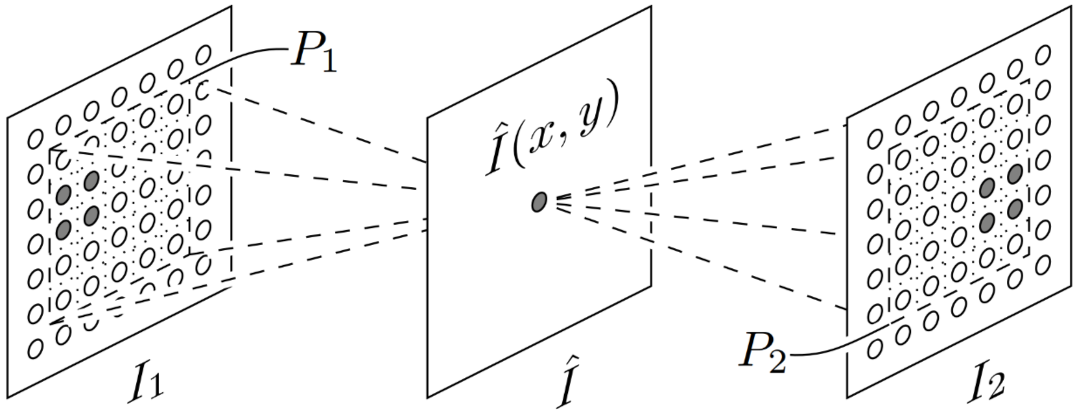
- Meyer et. al. addressed some shortcomings
  - oriented multi scale pyramid
  - propagate phase information across levels
  - bounded shift correction

S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung, “Phase-based frame interpolation for video”, CVPR 2015

Proposed Approach



(a) Interpolation by motion estimation and color interpolation

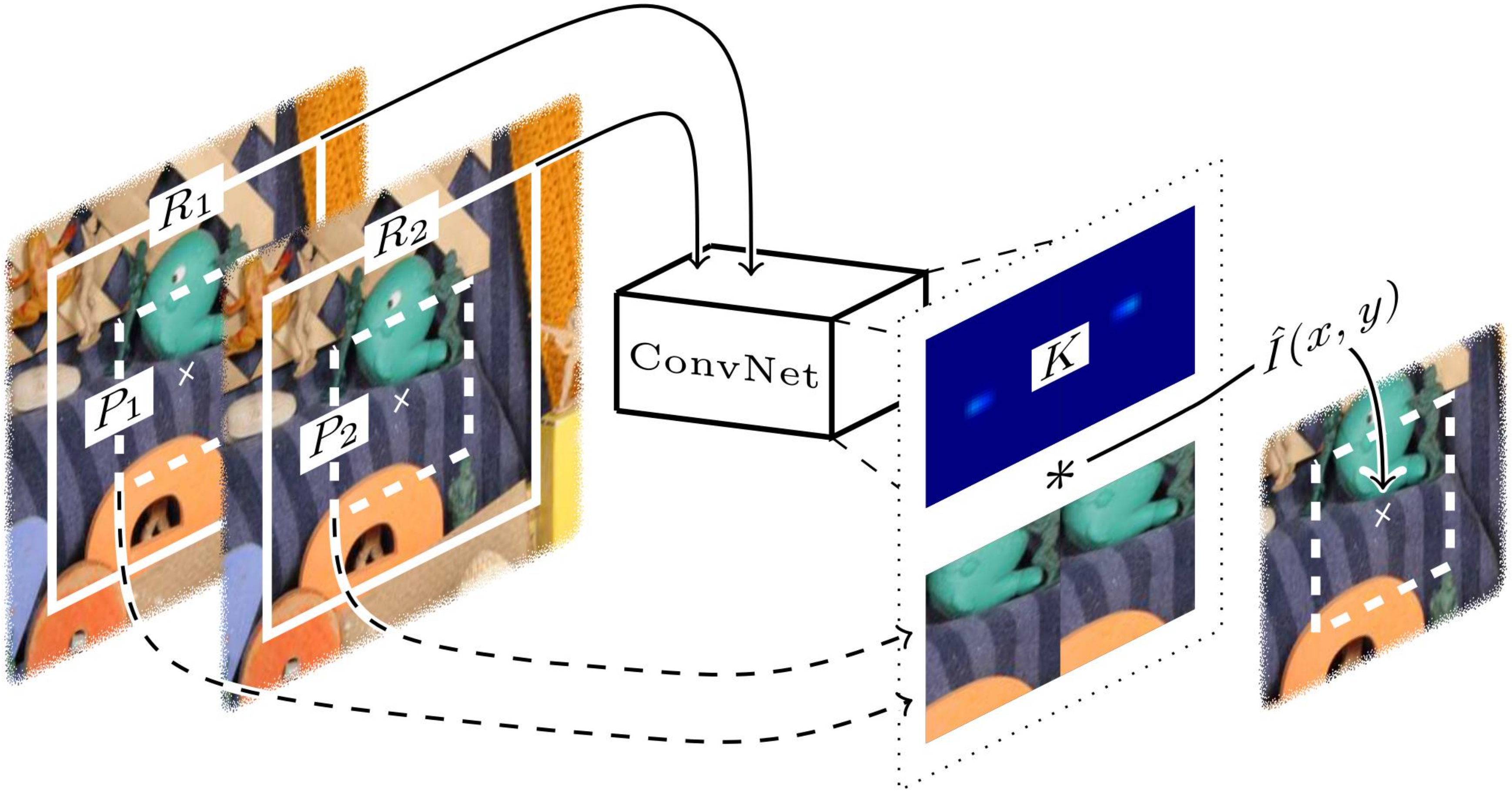


(b) Interpolation by convolution

Proposed Approach

- One kernel for each pixel
- 41 x 41 convolved patches centered at pixel
- 79 x 79 receptive field patches centered at pixel
    - handle aperture problem better
- Same kernel for all color channels
- Kernel entries are non zero and sum to 1

Proposed Approach



Architecture

type	BN	ReLU	size	stride	output
input	-	-	-	-	$6 \times 79 \times 79$
conv	✓	✓	7×7	1×1	$32 \times 73 \times 73$
down-conv	-	✓	2×2	2×2	$32 \times 36 \times 36$
conv	✓	✓	5×5	1×1	$64 \times 32 \times 32$
down-conv	-	✓	2×2	2×2	$64 \times 16 \times 16$
conv	✓	✓	5×5	1×1	$128 \times 12 \times 12$
down-conv	-	✓	2×2	2×2	$128 \times 6 \times 6$
conv	✓	✓	3×3	1×1	$256 \times 4 \times 4$
conv	-	✓	4×4	1×1	$2048 \times 1 \times 1$
conv	-	-	1×1	1×1	$3362 \times 1 \times 1$
spatial softmax	-	-	-	-	$3362 \times 1 \times 1$
output	-	-	-	-	$41 \times 82 \times 1 \times 1$

Training - Data

- 3000 Flickr videos
  - “driving”, “dancing”, “surfing”, “riding”, and “skiing”
- Scale to 1280 x 720
- Remove interlaced videos
- Triple frame groups of consecutive frames
- Extract random 150 x 150 patch
  - helps with augmentation
- Estimate optical flow between first and last patch[Tao12]

M. W. Tao, J. Bai, P. Kohli, and S. Paris, “SimpleFlow: A non-iterative, sublinear optical flow algorithm”, CGF 2012 9

Training – Data (cont’d)

- Pick **500K** groups according to the flow magnitude
  - Wide range of motion while avoiding small motion
- Pick **250K** groups with largest entropy
  - Remove low texture patches
- 10 % patches have flow magnitude > 20 pixels
  - mean flow of top 5 % is 25 pixels
  - largest flow is 38 pixels
- Data augmentation
  - random cropping
  - vertical and horizontal flipping
  - temporal order reversal

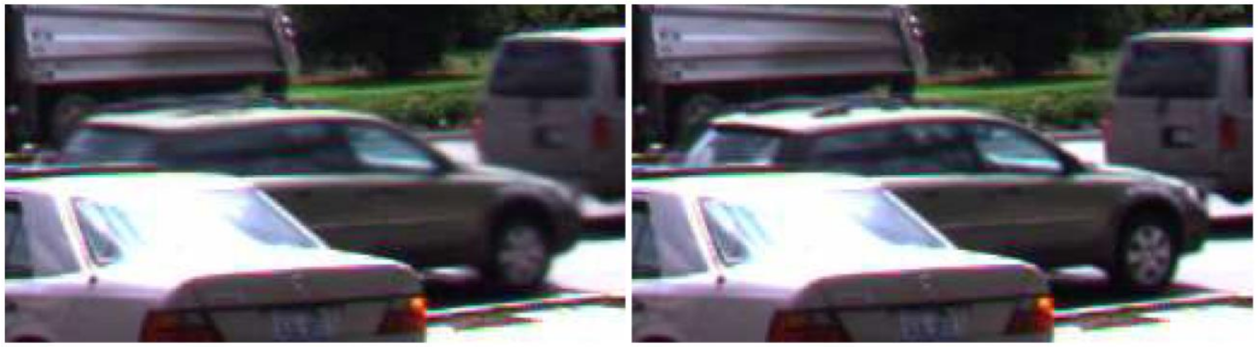
Training - Loss

- Color + Gradient Loss

$$E_c = \sum_i \| [P_{i,1} \; P_{i,2}] * K_i - \tilde{C}_i \|_1$$

$$E_g = \sum_i \sum_{k=1}^{8} \| [G_{i,1}^k \; G_{i,2}^k] * K_i - \tilde{G}_i^k \|_1$$

$$L = E_c + \lambda \cdot E_g$$



Implementation – Shift and Stitch

- Too slow and wasteful to process one pixel at a time
    - 921, 600 forward passes for a single 1280 x 720 image
    - 104 seconds per frame
    - highly redundant
- Entire image as input
    - non-adjacent and sparse output
- Slightly shifted versions of the image as input[Giusti13]
    - sparse results combined to construct the dense output
    - 64 forward passes for 64 shifted versions
    - 9 seconds per frame

A. Giusti, D. C. Cireşan, J. Masci, et. al "Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks", ICIP 2013

Implementation – Hyper parameters

- Kernel must be larger than the pixel motion
    - complexity increases with kernel size
    - enough to handle largest motion of 38 pixels in dataset
    - slightly larger to support resampling

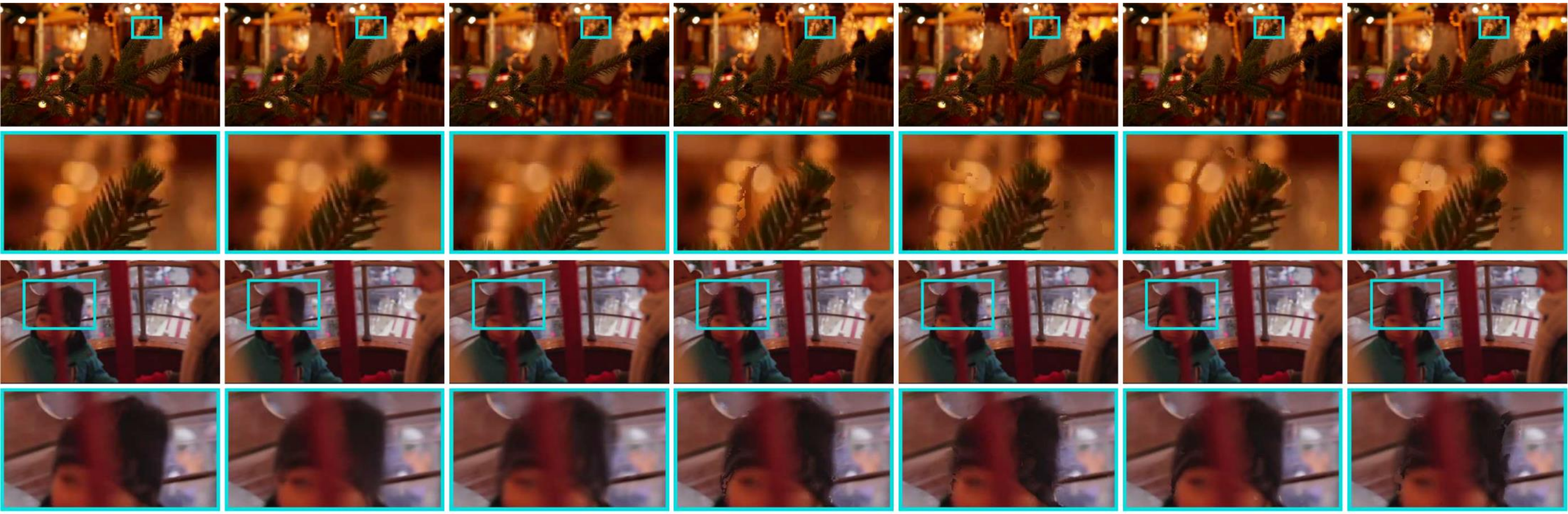
- Receptive field must be larger than convolutional kernel
    - sensitivity to motion decreases with increase in receptive field size
    - compromise chosen using validation dataset

Results - Middlebury

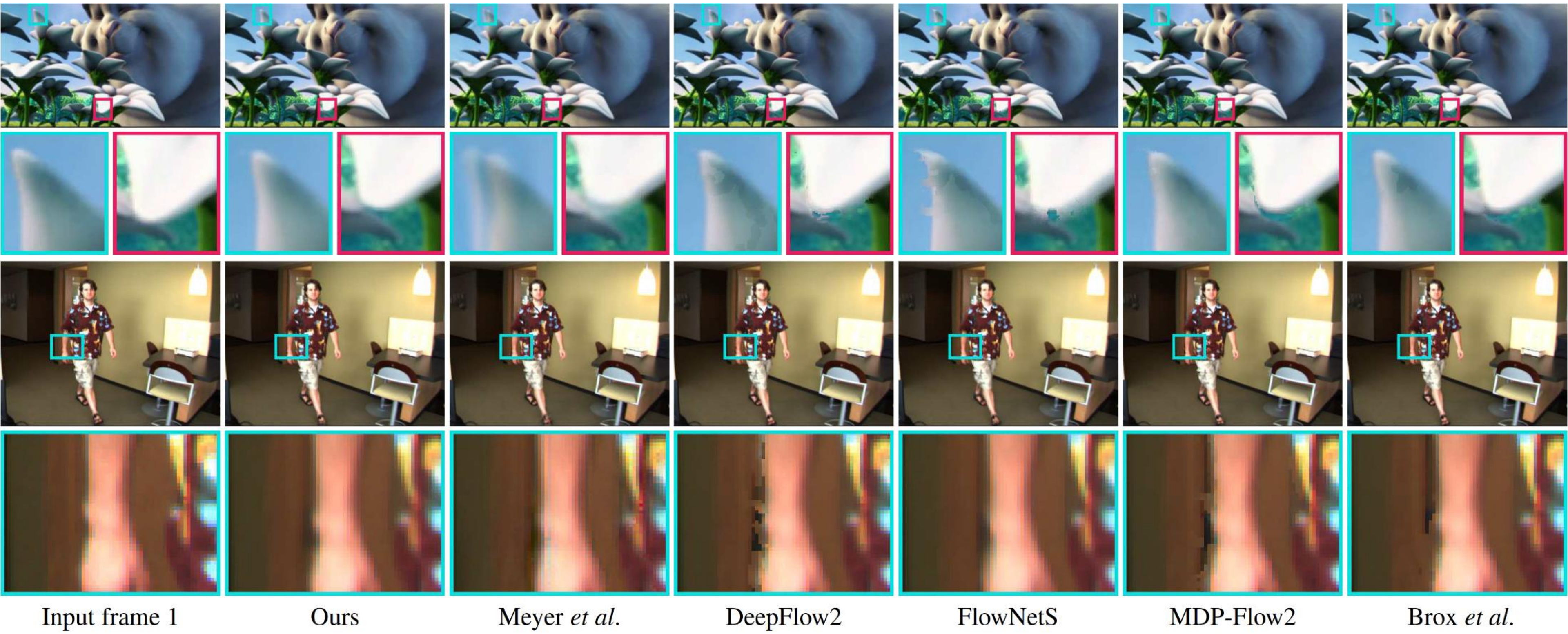
	Mequ.	Schef.	Urban	Teddy	Backy.	Baske.	Dumpt.	Everg.
Ours	3.57	4.34	5.00	6.91	**10.2**	**5.33**	**7.30**	**6.94**
DeepFlow2	2.99	3.88	**3.62**	5.38	11.0	5.83	7.60	7.82
FlowNetS	3.07	4.57	4.01	5.55	11.3	5.99	8.63	7.70
MDP-Flow2	**2.89**	**3.47**	3.66	**5.20**	**10.2**	6.13	7.36	7.75
Brox *et al.*	3.08	3.83	3.93	5.32	10.6	6.60	8.61	7.43

P. Weinzaepfel, J. Revaud, et. al, "DeepFlow: Large displacement optical flow with deep matching", ICCV 2013
A. Dosovitskiy, P. Fischer, E. Ilg, et. al, "FlowNet: Learning optical flow with convolutional networks", ICCV 2015
L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation", TPAMI 2012
T. Brox, A. Bruhn, et. al, "High accuracy optical flow estimation based on a theory for warping", ECCV 2004

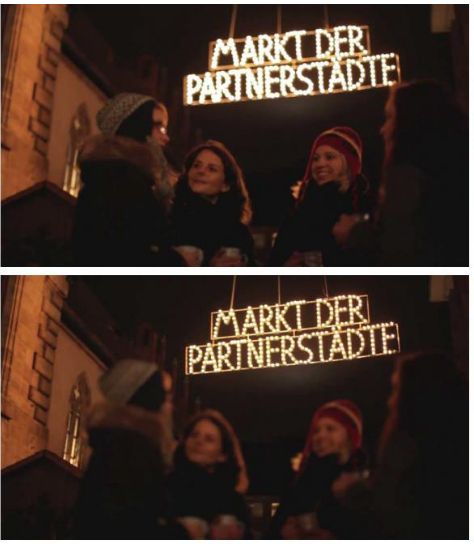
Results - Blur



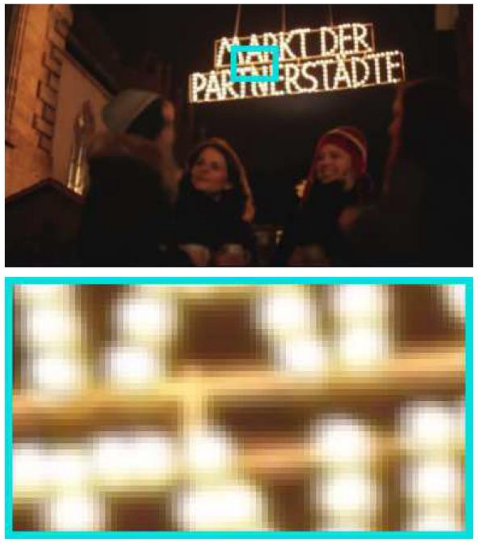
Results - Occlusion



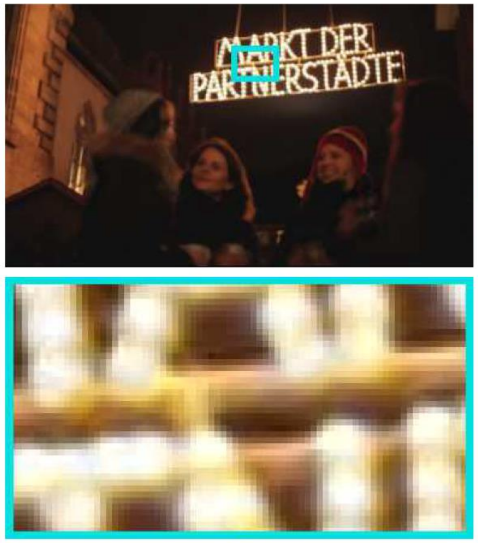
Results – Illumination Change



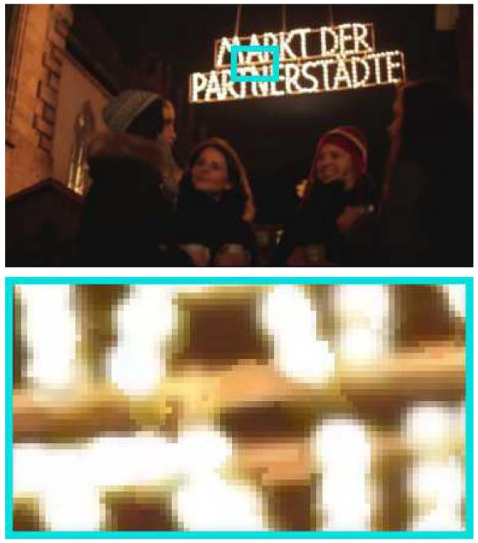
Input frames



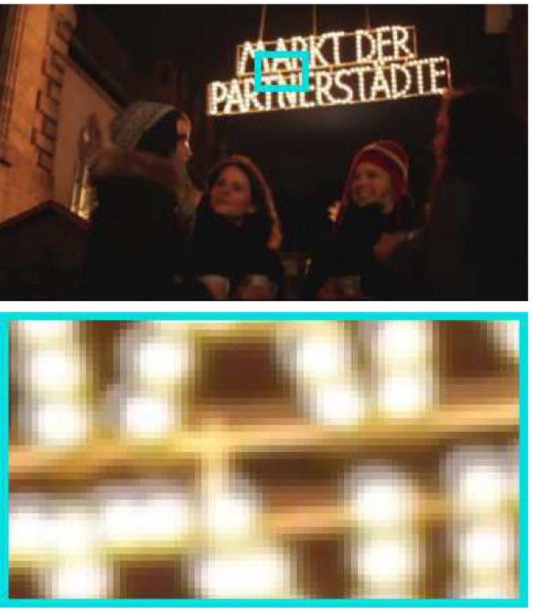
Ours



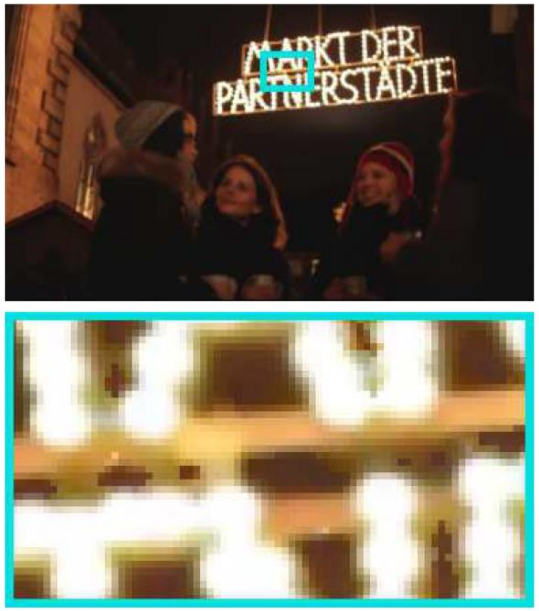
Meyer *et al.*



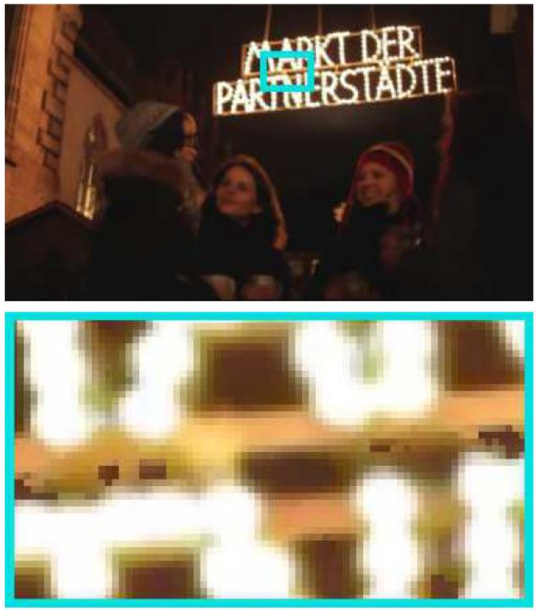
DeepFlow2



FlowNetS



MDP-Flow2

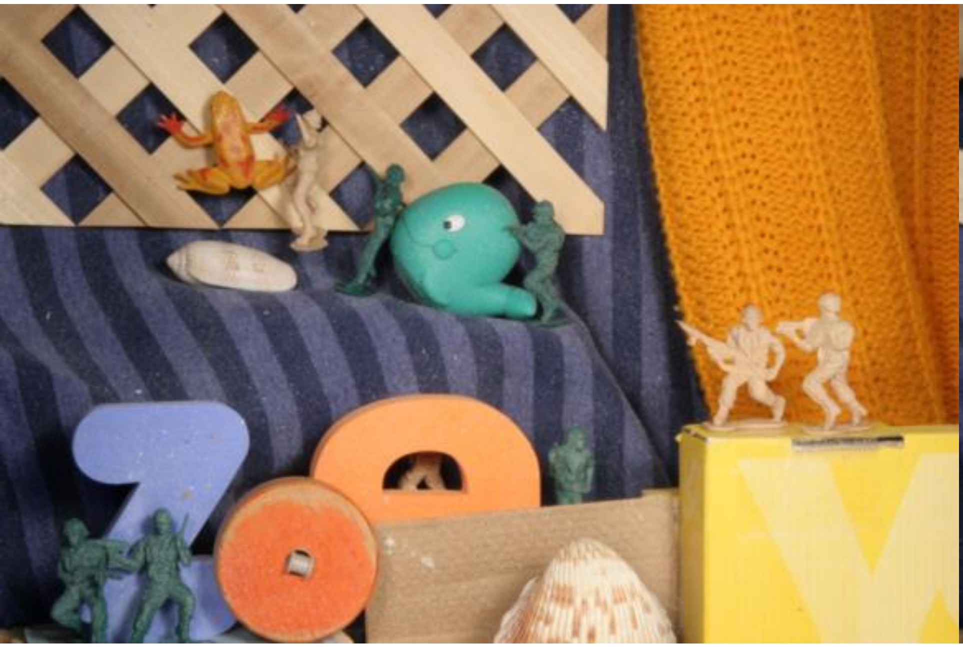


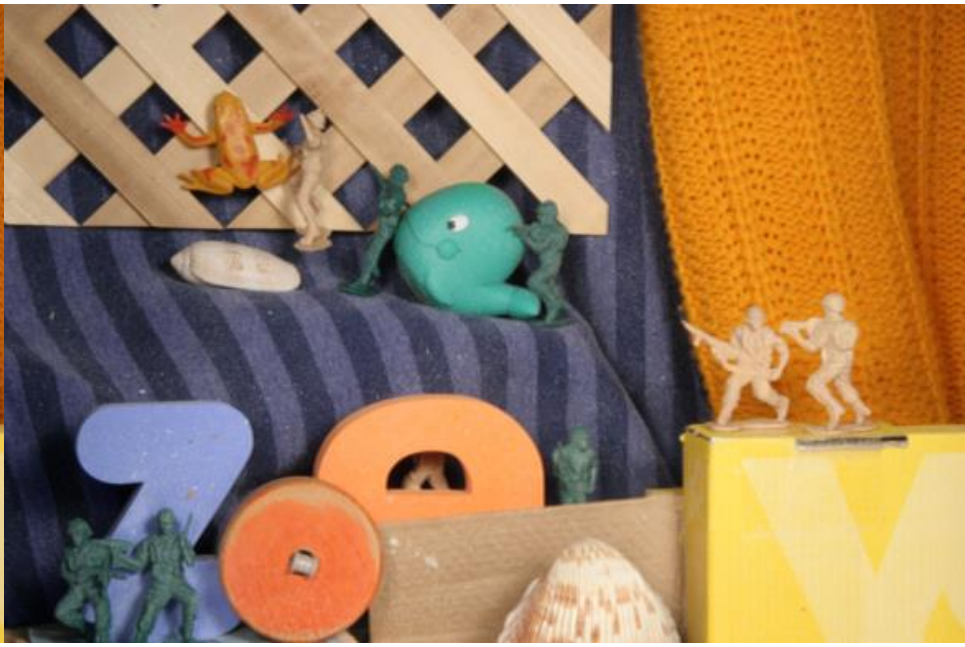
Brox *et al.*

Results – Phase Interpolation

CNN

Phase





Frame difference 1

Results – Phase Interpolation

CNN

Phase

Frame difference 2

Results – Phase Interpolation

CNN

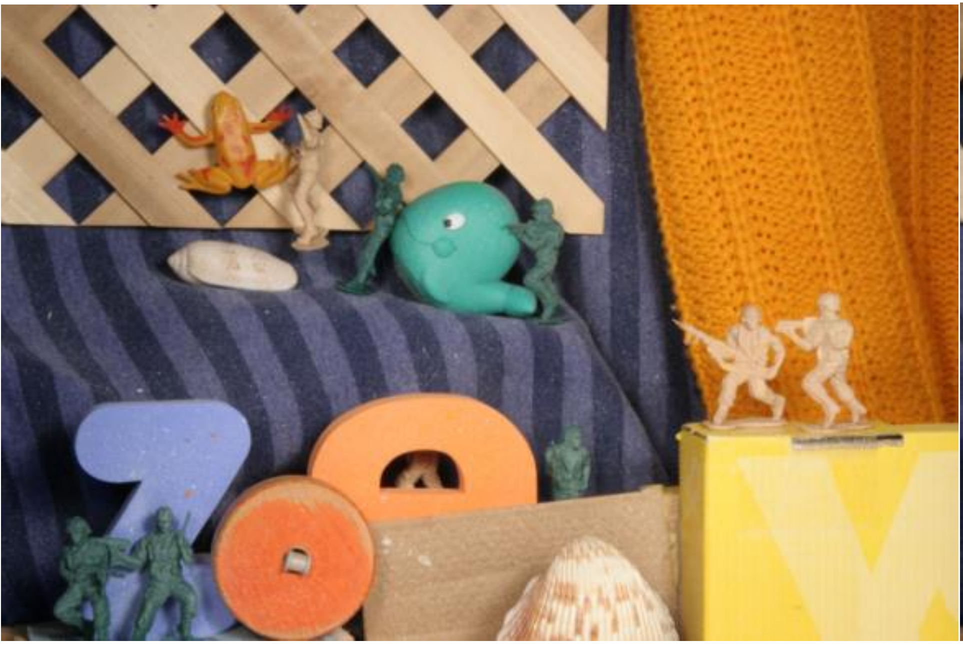
Phase

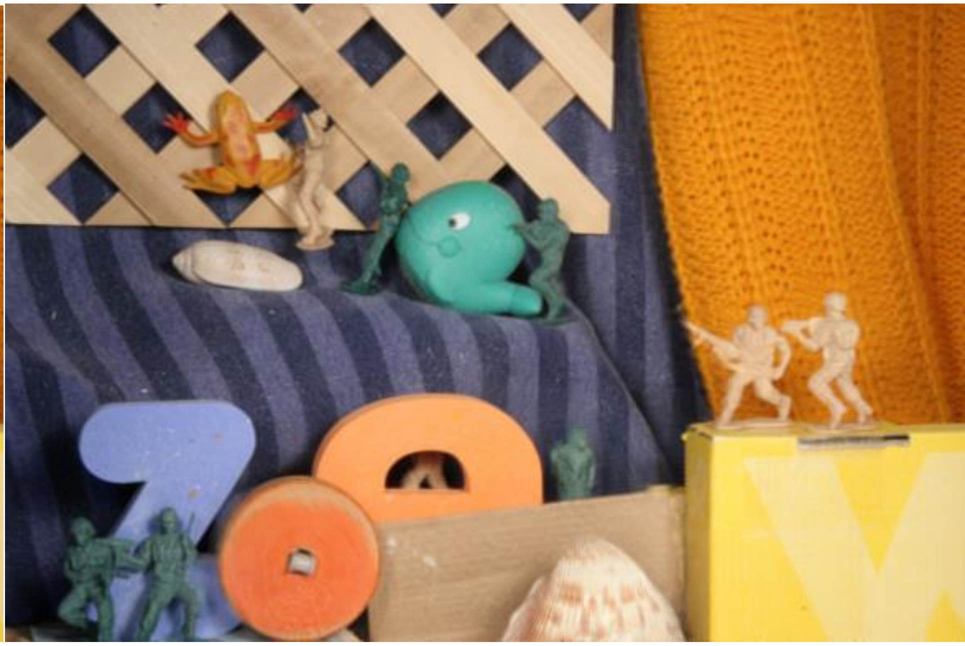
Frame difference 3

Results – Phase Interpolation

CNN

Phase



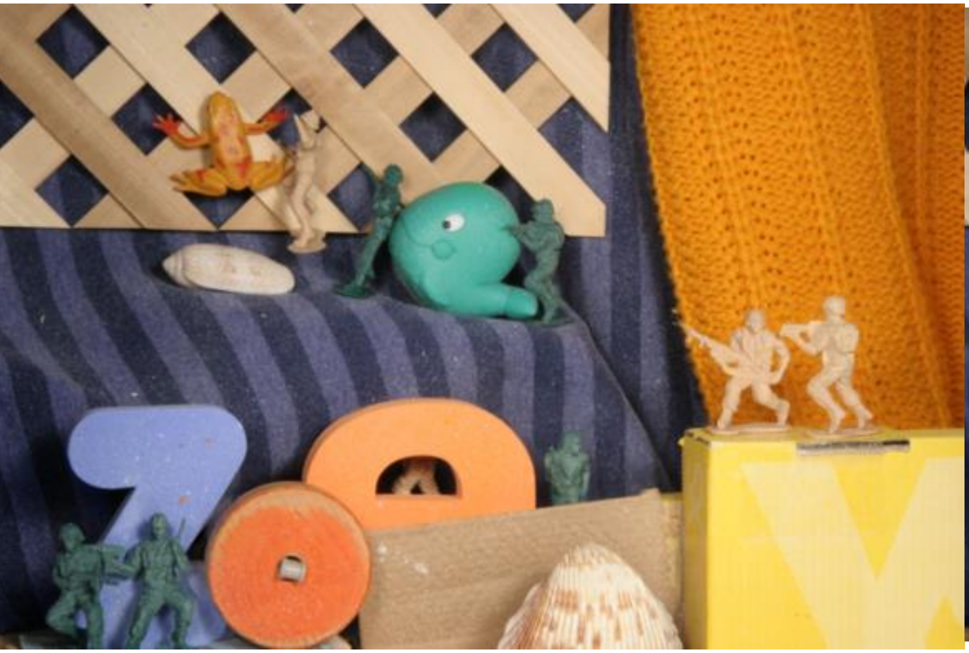


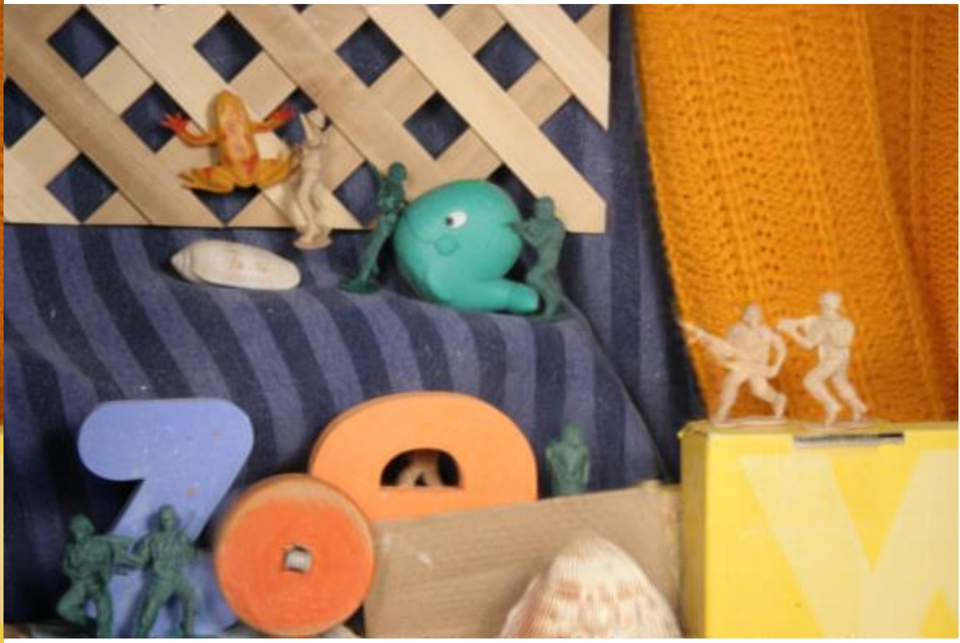
Frame difference 4

Results – Phase Interpolation

CNN

Phase





Frame difference 5

Results – Phase Interpolation

- More results comparing AdaConv with Phase based interpolation on Middlebury dataset are available here:

https://drive.google.com/open?id=1xFt0HL76xTub706j5i2Nm4UrDdUfrnNd

Results – Sintel (**2X** interpolation)

Results – Sintel (**4X** interpolation)

Results – Sintel (**8X** interpolation)

Results – Sintel (**16X** interpolation)

Results – Sintel

- More results for 2x to 32x interpolation of Sintel test sequences using AdaConv are available here:

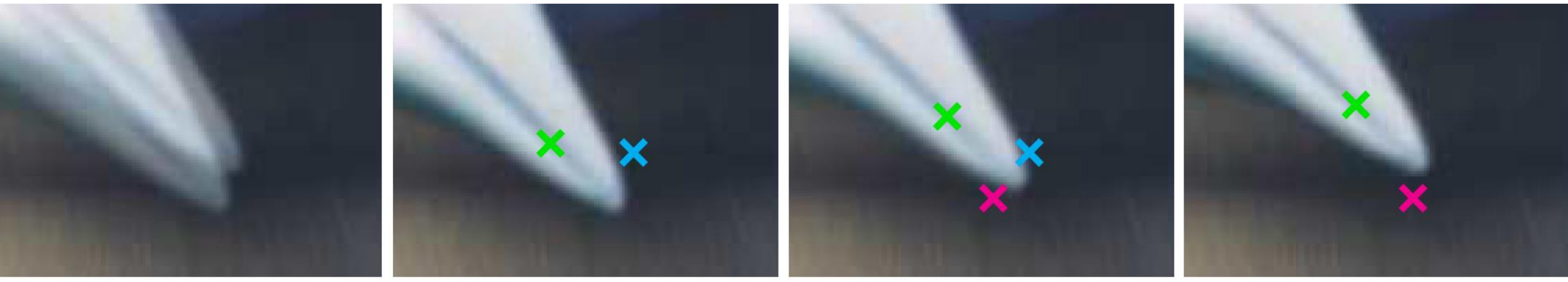
https://drive.google.com/open?id=1t4wpqAaKyEJjuF39q3VopGA1gTgn7nPX

Advantages

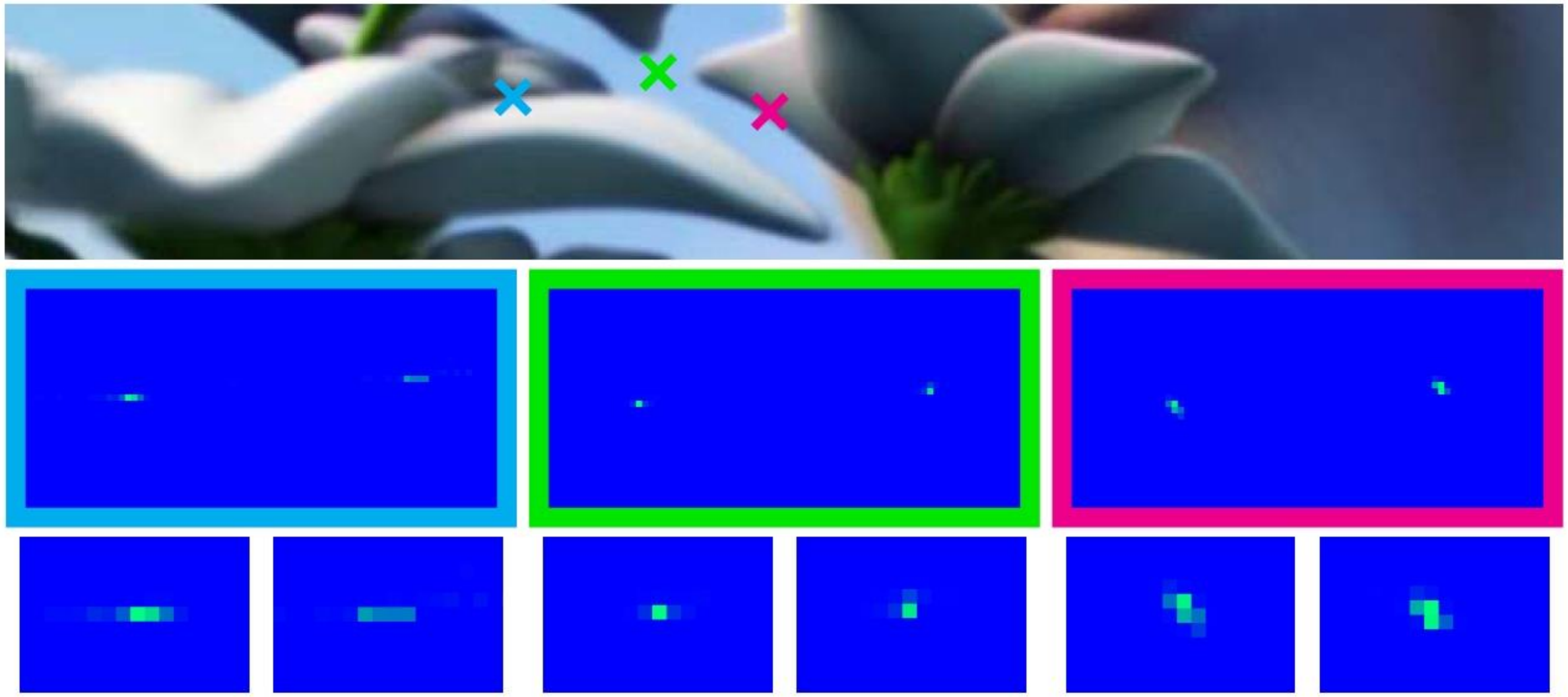
- Combination into single step is more robust
- Accounts for difficult cases like occlusion
- Edge aware filtering to give sharper results

Advantages – Occlusion Handling

- Green: Visible in both images
  - both sub kernels non zero
- Red: Visible only in Image 2
  - sub kernel 1 is almost zero
- Cyan: Visible only in Image 1
  - sub kernel 2 is almost zero



Advantages – Edge Aware Interpolation

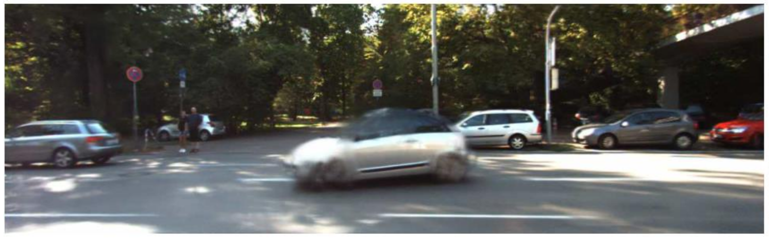


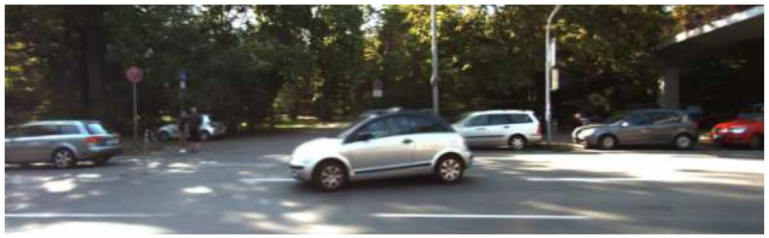
Limitations

- Cannot handle motion larger than 41 pixels

(a) Left view

(b) Right view


(c) Ours - full resolution


(d) Ours - half resolution

- Cannot handle arbitrary interpolation ratio

Questions ?