CNN Based Pipeline for Optical Flow

Tal Schuster, June 2017

Based on:

PatchBatch: a Batch Augmented Loss for Optical Flow, (Gadot, Wolf) CVPR 2016

Optical Flow Requires Multiple Strategies (but only one network), (Schuster, Wolf, Gadot) CVPR 2017

Overview

Goal – Get SOTA results in main optical flow benchmarks

Was done by:

- Constructing a Deep Learning based pipeline (modular)
- Architectures exploration
- Loss function augmentations
- Per-batch statistics
- Learning methods

Problem Definition

Problem Definition - Optical Flow

Given 2 images, compute a dense Optical Flow Field describing the motion between both images (i.e. **pure** optical flow):

2 X (h,w,1 / 3) ⟶ (h,w,2)

Where:

- h - image height, w - image width
- (h,w,1 / 3) - a grayscale or RGB image
- (h,w,2) - a 3D tensor describing for each point (x,y) in image-A a 2D-flow vector: $(\Delta x, \Delta y)$

Accuracy measures:

- Based on GT (synthetic or physically obtained) - KITTI, MPI-Sintel
- F_err - % of pixels with *euclidean error* > z pixels (usually z=3)
- Avg_err - *mean* of euclidean errors over all pixels

DB - KITTI2012

- LIDAR based
- ~50% coverage

DB - KITTI2012

Error	Out-Noc	Out-All	Avg-Noc	Avg-All
2 pixels	10.51 %	24.65 %	1.2 px	5.8 px
3 pixels	7.28 %	20.76 %	1.2 px	5.8 px
4 pixels	5.41 %	18.01 %	1.2 px	5.8 px
5 pixels	3.97 %	15.69 %	1.2 px	5.8 px

DB - KITTI2015

Error	Fl-bg	Fl-fg	Fl-all
All / All	8.45	33.93	11.29
All / Est	8.45	33.93	11.29
Noc / All	3.38	33.93	7.16
Noc / Est	3.38	33.93	7.16

DB - MPI SINTEL

- Synthetic (computer graphics)
- ~100% coverage

Solutions

Traditional computer vision methods

- Global constraints (Horn-Schunk, 1981) – Brightness constancy + smoothness asm.
- Local constraints (Lucas-Kanade, 1981)

Main disadvantage – small objects and fast movements

Descriptor based methods

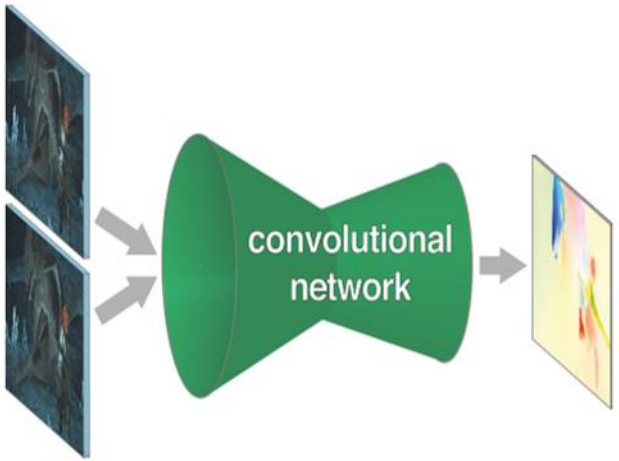
- Sparse to dense (Brox-Malik, 2010)

Descriptors

SIFT, SURF, HOG, DAISY, etc. (handcrafted)

CNN methods

- End to End – Flownet (Fischer et al., 2015)



Reference Work – Zbontar & Lecun, 2015

Solving Stereo-Matching vs. Optical Flow

Classification-based vs. metric learning

To compute the classification, the network needs to observe both patches simultaneously

Component	Runtime
Convolutional neural network	95 s
Semiglobal matching	3 s
Cross-based cost aggregation	2 s
Everything else	0.03 s

Table 2. Time required for prediction of each component.

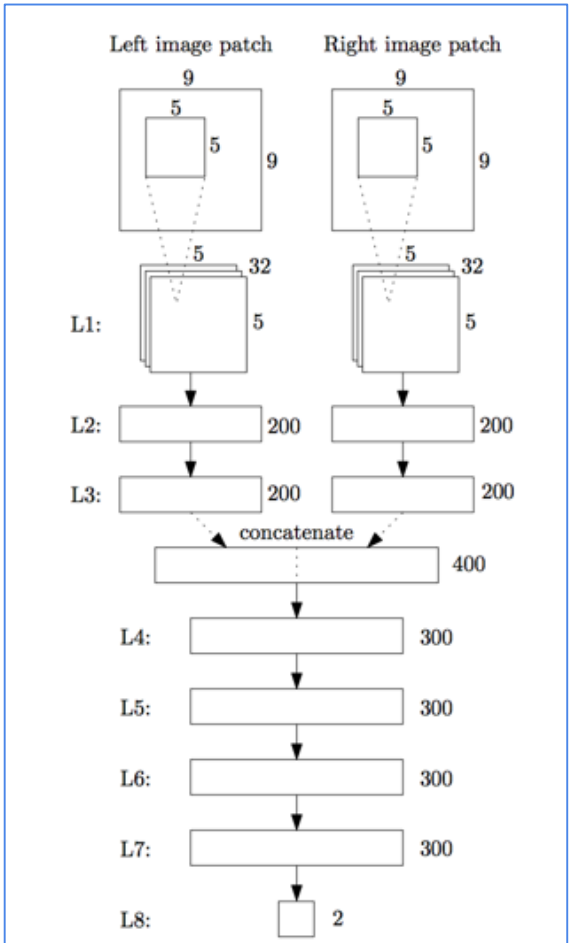


Figure 2. The architecture of our convolutional neural network.

The PatchBatch pipeline

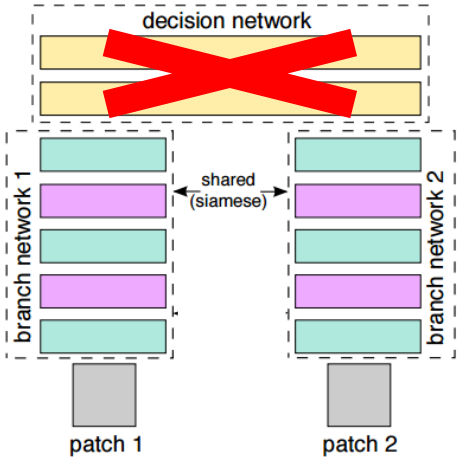
PatchBatch - DNN

Siamese DNN - i.e., tied weights due to symmetry

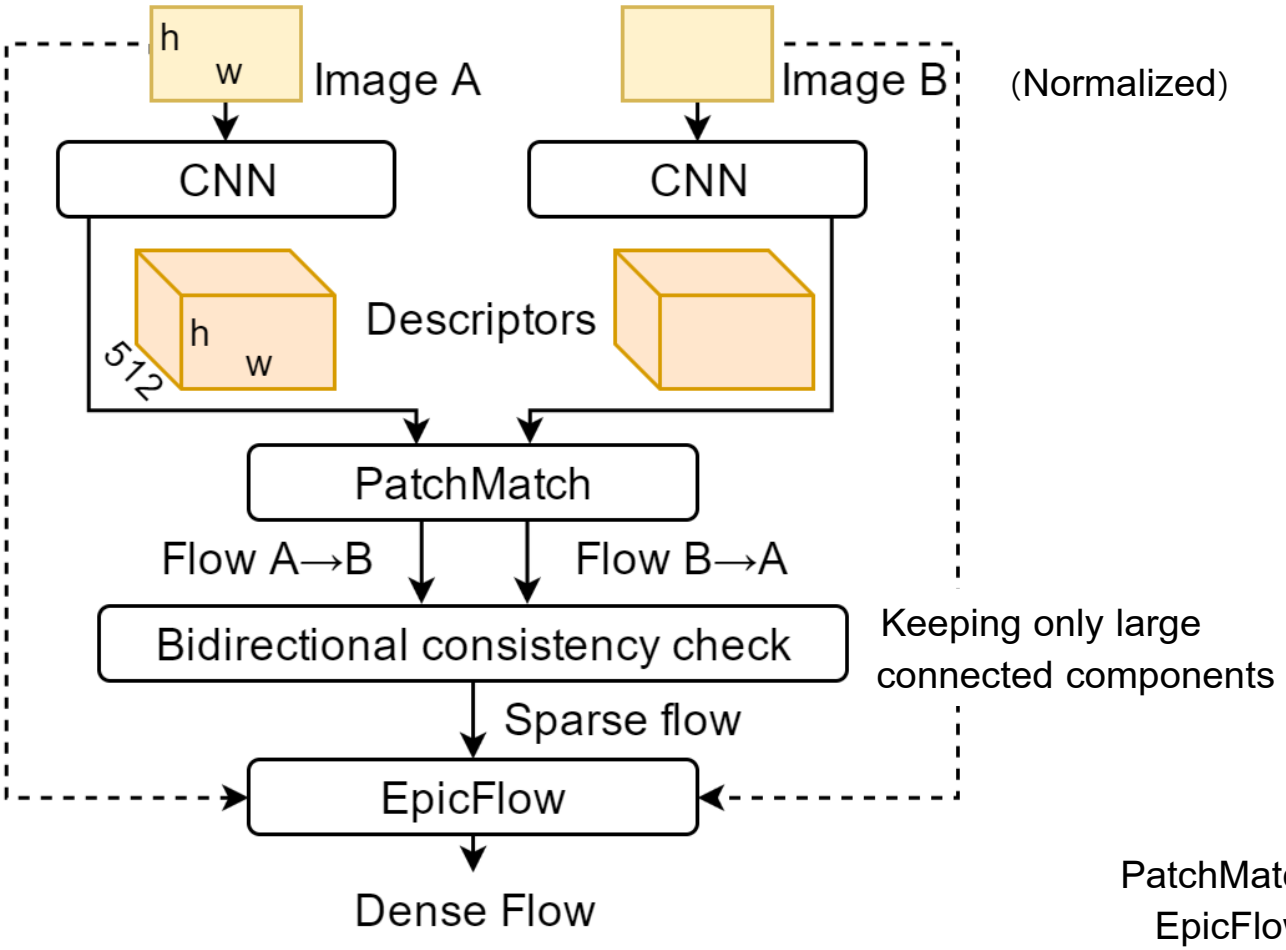
Leaky ReLU

Should be FAST:

- Matching function = **L2**
- Conv only
- Independent descriptor computation



PatchBatch - Overall Pipeline



PatchMatch - Barnes et al. 2010
EpicFlow - Revaud et at. 2015

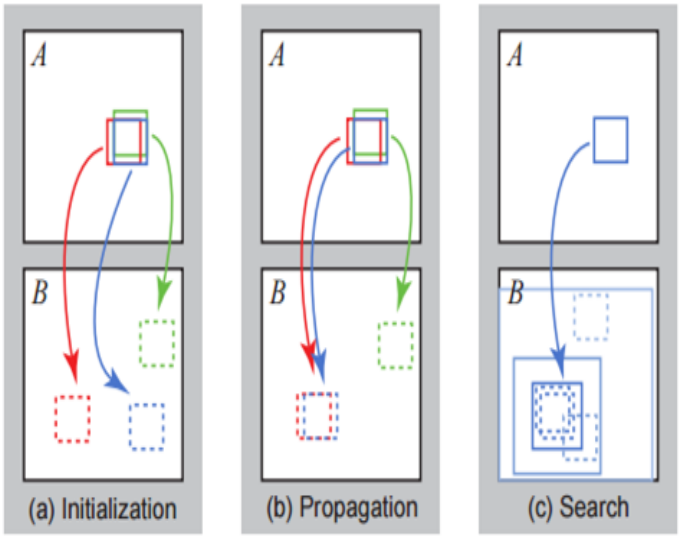
PatchBatch - ANN

PatchMatch:

(Descriptors, Matching function) → ANN

ANN and not ENN : O(N^2) → O(N*logN)

2 iterations are enough



1. Initialization (random)
2. Propagation

$$f(x,y) = argmin\{D(f(x,y)), D(f(x-1,y)), D(f(x,y-1))\}$$

(+1 on even iterations)

3. Search

$$u_i = v_0 + w\alpha^i R_i$$

4. Return to step 2

$R_i \in [-1,1] \times [-1,1]$

w - max radius

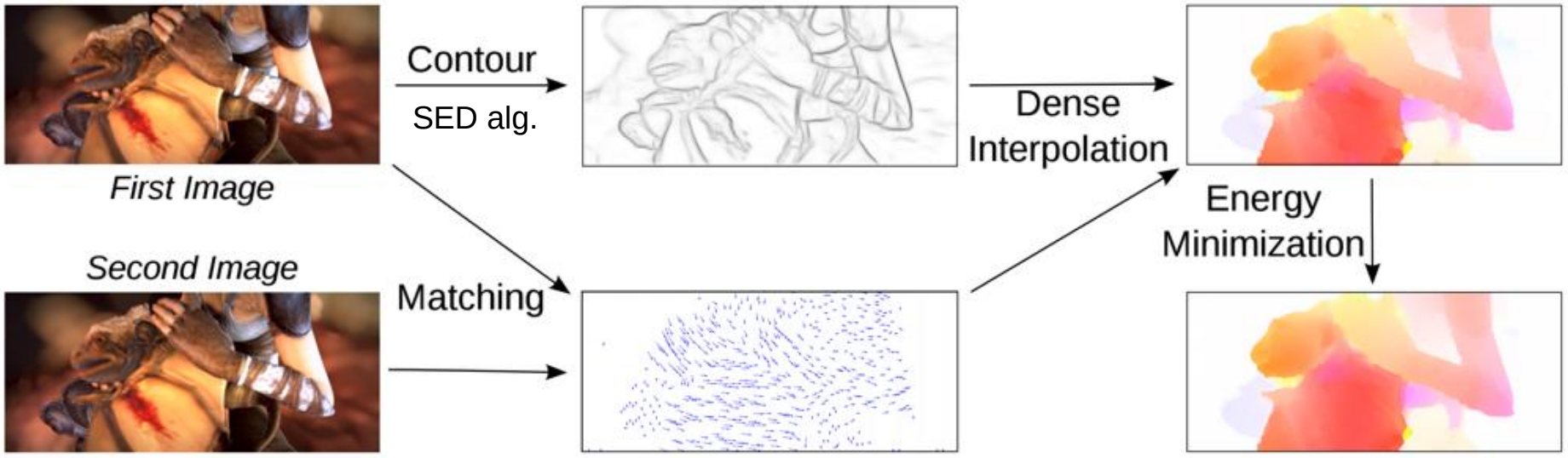
α - step ($= \frac{1}{2}$)

PatchBatch - Post-Processing

EpicFlow (Edge-Preserving Interpolation of Correspondences)

Sparse -> Dense

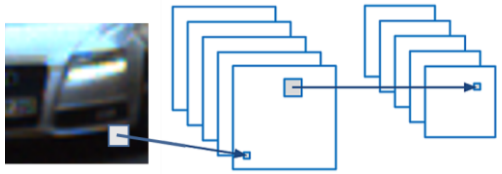
Average support affine transformations based on geodesic distance on top of edges map



PatchBatch - CNN

Layer	Filter/Stride	Output size
Input	–	$1 \times 51 \times 51$
Conv1	$3 \times 3 / 1$	$32 \times 49 \times 49$
Batch Normalization	–	$32 \times 49 \times 49$
Max Pool	$2 \times 2 / 2$	$32 \times 25 \times 25$
Conv2	$3 \times 3 / 1$	$64 \times 23 \times 23$
Batch Normalization	–	$64 \times 23 \times 23$
Max Pool	$2 \times 2 / 2$	$64 \times 12 \times 12$
Conv3	$3 \times 3 / 1$	$128 \times 10 \times 10$
Batch Normalization	–	$128 \times 10 \times 10$
Max Pool	$2 \times 2 / 2$	$128 \times 5 \times 5$
Conv4	$3 \times 3 / 1$	$256 \times 3 \times 3$
Batch Normalization	–	$256 \times 3 \times 3$
Max Pool	$2 \times 2 / 2$	$256 \times 2 \times 2$
Conv5	$2 \times 2 / 1$	$512 \times 1 \times 1$
Batch Normalization	–	$512 \times 1 \times 1$

Table 1. The network model for representing a grayscale 51×51 input patch as $512D$ vector. The Batch Normalization is our fine-grained variant. Leaky ReLU units [26] (with $\alpha = 0.1$) are used as activation functions following the five batch normalization layers.



Batch Normalization-

Solves the “internal covariate shift” problem

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Per pixel instead of per feature map

PatchBatch - Loss

- DrLIM - Dimensionality Reduction by Learning an Invariant Mapping (LeCun, 2006)

Orig DrLIM (SPRING)

$$(1-Y)\frac{1}{2}D_w^2+(Y)\frac{1}{2}\{\max(0,m-D_w)\}^2$$

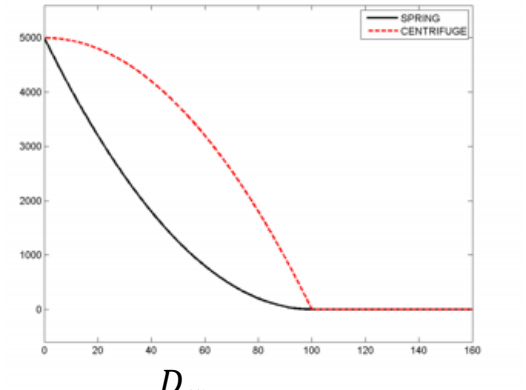
CENT

$$(1-Y)\frac{1}{2}D_w^2+(Y)\frac{1}{2}\{\max(0,m^2-D_w^2)\}$$

CENT+SD

$$(1-Y)\lambda D_w^2+(Y)\lambda\{\max(0,m^2-D_w^2)\}+(1-\lambda)(\sigma_0+\sigma_1)$$

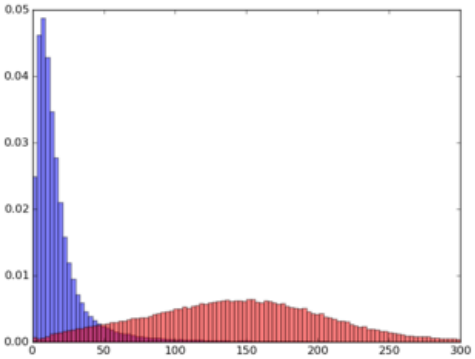
Negative pairs



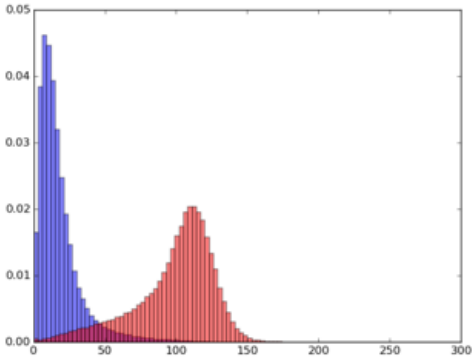
L

D_w

(a)



(b) CENT

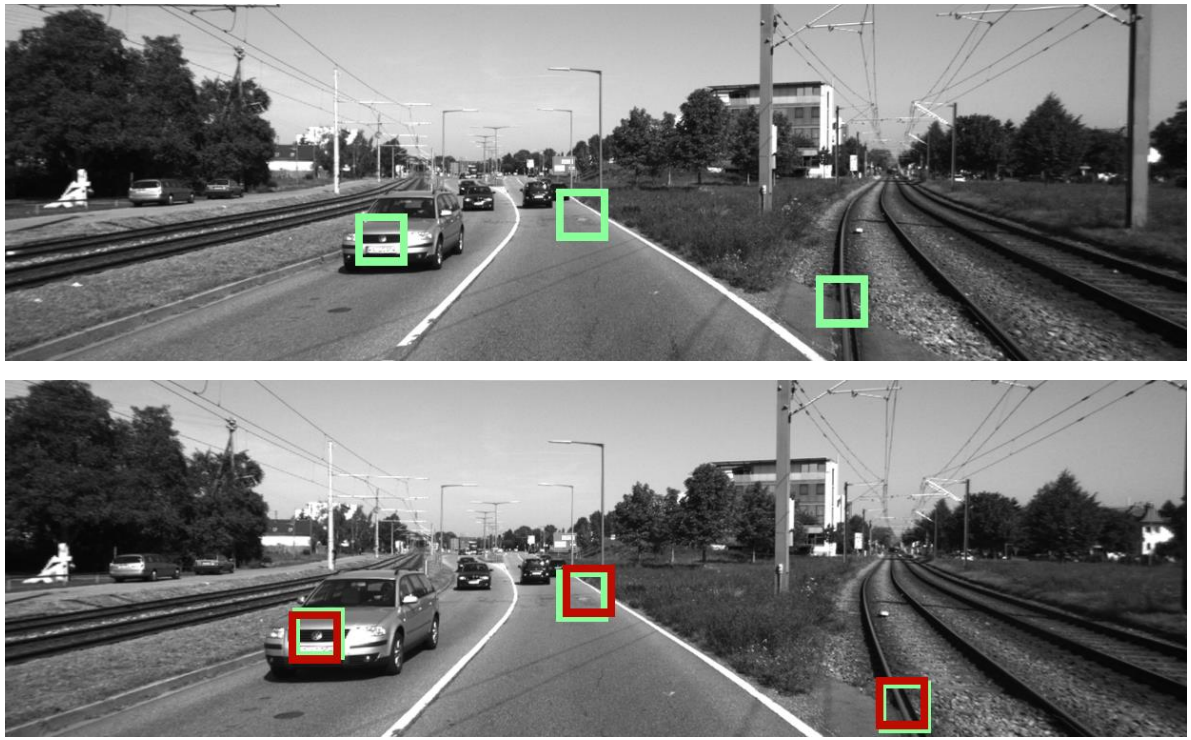


(c) CENT + SD

PatchBatch - Training Method

Negative sample – random 1-8 pixels from the true match

Data augmentation - flipping , rotating 90°



Results

Benchmarks

Method	Out-Noc	Running time
PatchBatch-ACCRTE-PS71	5.29%	60.5s
PatchBatch-ACCURATE	5.44%	50.5s
PH-Flow [39]	5.76%	800s
FlowFields [1]	5.77%	23s
CPM-Flow (anon.)	5.80%	2s
NLTGV-SC [30]	5.93%	16s
PatchBatch-FAST	5.94%	25.5s
DDS-DF [37]	6.03%	1m
TGV2ADCSIFT [5]	6.20%	12s
DiscreteFlow [28]	6.23%	3m

Method	Fl-all	Running time
PatchBatch-ACCURATE	21.69%	50.5s
DiscreteFlow [28]	22.38%	3min
CPM-Flow (anon.)	24.24%	2s
EpicFlow [32]	27.10%	15s
FilteringFlow (anon.)	28.50%	116s
DeepFlow [38]	29.18%	17s
HS [35]	42.18%	2.6m
DB-TV-L1 [40]	47.97%	16s
HAOF [6]	50.29%	16.2s
PolyExpand [14]	53.32%	1s

Table 5. Top 10 KITTI2015 Pure Optic Flow Algorithms as of the submission date. Fl-all is the percentage of pixels with euclidean error > 3 pixels. The FAST network was not trained on this benchmark by the submission time.

Method	EPE all, 'final' pass
FlowFields [1]	5.810
CPM-Flow (anon.)	5.960
DiscreteFlow [28]	6.077
EpicFlow [32]	6.285
Deep+R [13]	6.769
PatchBatch-CENT+SD	6.783
DeepFlow2 (anon.)	6.928
PatchBatch-SPRG	7.188
SparseFlowFused [36]	7.189
DeepFlow [38]	7.212
FlowNetS+ft+v [15]	7.218
NNF-Local [9]	7.249
PatchBatch-SPRG+SD	7.281
PatchBatch-CENT	7.323
SPM-BP [25]	7.325
AggregFlow [16]	7.329

Table 6. Top MPI-Sintel results as of the submission date. Each number represents the EPE (end-point-error), averaged over all the pixels in the comparison images, using the 'final' rendering pass of MPI-Sintel. Four ACCURATE variants are shown. The CENT-FIGURE+SD network is ranked 6th as of the paper's submission date. The FAST network was not trained on this benchmark by that date. The TF+OFM method [22] (EPE 6.727) is removed from this table since it is not a pure optical flow method.

How can we Improve the Results?

Architecture Modifications

PatchBatch - CNN

Increased Patch and Descriptor sizes

Layer	Filter/Stride	Output size (51)	Output size (71)
Input	–	$1 \times 51 \times 51$	$1 \times 71 \times 71$
Conv1	$3 \times 3 / 1$	$32 \times 49 \times 49$	$32 \times 69 \times 69$
Batch Normalization	–	$32 \times 49 \times 49$	$32 \times 69 \times 69$
Max Pool	$2 \times 2 / 2$	$32 \times 25 \times 25$	$32 \times 35 \times 35$
Conv2	$3 \times 3 / 1$	$64 \times 23 \times 23$	$64 \times 33 \times 33$
Batch Normalization	–	$64 \times 23 \times 23$	$64 \times 33 \times 33$
Max Pool	$2 \times 2 / 2$	$64 \times 12 \times 12$	$64 \times 17 \times 17$
Conv3	$3 \times 3 / 1$	$128 \times 10 \times 10$	$128 \times 15 \times 15$
Batch Normalization	–	$128 \times 10 \times 10$	$128 \times 15 \times 15$
Max Pool	$2 \times 2 / 2$	$128 \times 5 \times 5$	$128 \times 8 \times 8$
Conv4	$3 \times 3 / 1$	$256 \times 3 \times 3$	$256 \times 6 \times 6$
Batch Normalization	–	$256 \times 3 \times 3$	$256 \times 6 \times 6$
Max Pool	$2 \times 2 / 2$	$256 \times 2 \times 2$	$512 \times 3 \times 3$
Conv5	$2 \times 2 / 1$	$512 \times 1 \times 1$	$512 \times 2 \times 2$
Batch Normalization	–	$512 \times 1 \times 1$	$512 \times 2 \times 2$
Reshape Layer	–	–	$2048 \times 1 \times 1$

Table 3.1: Network models for representing a grayscale 51×51 input patch as $512D$ vector and a 71×71 patch as 2048 vector. Leaky ReLU units [28] (with $\alpha = 0.1$) are used as activation functions following the five batch normalization layers. All reported results on this work are using the ACCURATE Batch Normalization version as described in the PatchBatch paper.

Hinge Loss with SD

- **Hinge Loss** instead of DrLIM
- Trained on **Triplets** - <A, B-match, B-non-match >
- Keeping the additional **SD component**

$$L_H = \frac{1}{n}\sum_{i=1}^{n} max(0, m + D_{i,match} - D_{i,non-match})$$

$$L_{H+SD} = \lambda L_H + (1-\lambda)(\sigma_{D_{match}} + \sigma_{D_{non-match}})$$

$$m = 100, \lambda = 0.8$$

$$n = 50k$$

Failed Attempts

Data augmentation

- Rotations (random $+/-\ \alpha$)
- Colored patches (HSV or other decomposition)

Loss function

- Foursome (A, B, A', B')

  A, B – matching patches

  A'– Patch from Image A that is closest to B

  $H(A,B) = \max(0, m - L_2(A,B))$

  $L = L_2(A,B) + I(B \neq B') * H(A,B') + I(A \neq A') * H(A',B)$

Sample Mining

- PatchMatch output
- Patches distance
- Descriptor distance

Optical Flow as a Multifaceted Problem

Success of methods

The challenge of **large displacements**

KITTI 2015 average error:

- Foreground – 26.43 %
- Background – 11.43 %

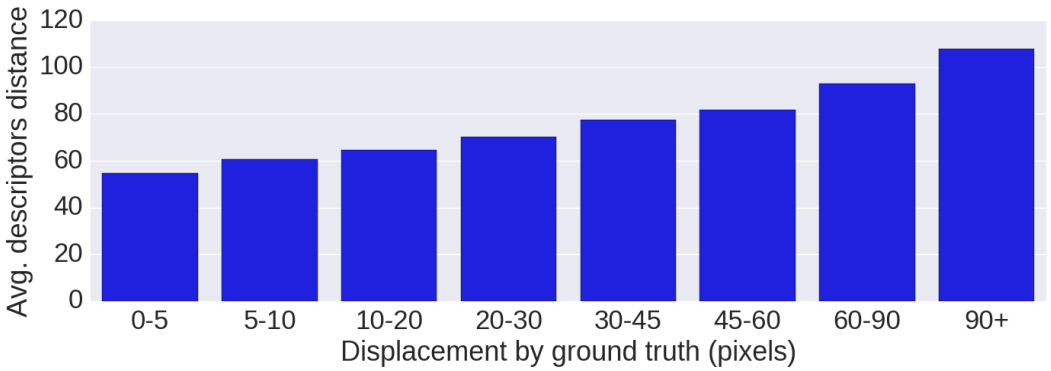
Possible causes:

- Matching algorithm
- Descriptors quality

MPI-Sintel results table

s0-10	s10-40	s40+
1.083	3.227	39.409
1.135	3.727	38.021
1.152	3.986	39.985
0.800	2.856	45.063
1.512	3.765	39.761
1.157	3.837	41.687
0.725	3.064	45.858

PatchBatch on KITTI 2012 – distance between true matches



Descriptors Evaluation

Defined a quality measurement of descriptors for matching

d_p is a **distractor** of pixel p if the L_2 distance between d_p and p is lower than the L_2 distance of p with its matching pixel descriptor.

Counted up to 25 pixels from the examined pixel.

Distractors by displacement

Amount of distractors increase with displacement range

Goal: improve results for large displacements without reducing for other ranges.

Train set	0-5	5-10	10-20	20-30	30-45	45-60	60-90	90-∞
Baseline (all)	**2.32**	7.32	5.32	9.38	25.21	50.43	67.32	216.39

Distractors by displacement

Amount of distractors increase with displacement range

Goal: improve results for large displacements without reducing for other ranges.

Expert models

- Training only on sub ranges
- Improving results for large displacements is possible
- Implies the **need of different features** for **different patches**

Train set	0-5	5-10	10-20	20-30	30-45	45-60	60-90	90-∞
Baseline (all)	**2.32**	7.32	5.32	9.38	25.21	50.43	67.32	216.39
<30	2.46	**6.91**	**5.25**	**8.57**	26.39	51.76	65.15	209.40
>30	3.03	9.07	5.64	10.29	**24.74**	**46.81**	**56.69**	**199.61**

Learning with Varying Difficulty

Gradual Learning Methods

Deal with varying difficulty

Curriculum (Bengio et al. 2009):

- Samples are pre-ordered
- Curriculum by displacement
- Curriculum by distance (of false sample)

Self-Paced (Kumar et al. 2010):

- No need to pre-order
- Sample hardness increases with time (by loss value)

Hard Mining (Simo-Serra et al. 2015):

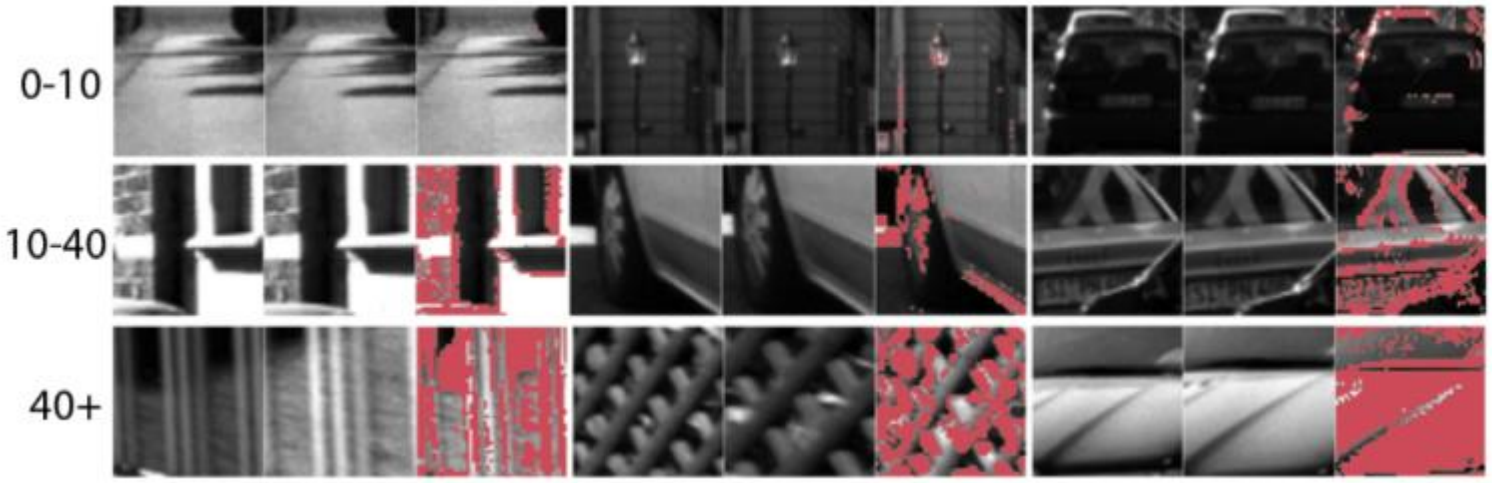
- Backpropagate only some ratio of harder samples
- Used for training local descriptors with triplets

All methods did not improve match over baseline – Why?

Need for variant extracting strategies

Large motions are mostly correlated with more changes in appearance:

1. Background changes
2. View point changes -> occluded parts
3. Distance and angle to light source -> illumination
4. Scale (when moving along the Z-axis)



Learning for Multiple Strategies and Varying Difficulty

Our Interleaving Learning Method

Goal: Deal with **multiple sub-tasks**

Learning ML models

- Mostly in random order (SGD)
- Applying gradual methods can effect randomness

Interleaving Learning

- **Maintaining the random order of categories while adjusting the difficulty**

Motivated by psychological research (Kornell - Bjork)

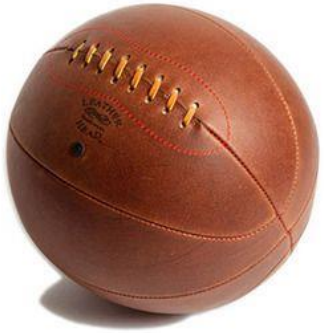
- Massing vs. Interleaving
- Experiments on classification tasks, sports, etc.

Classification: Painting to Artist

Learning Concepts and Categories – Kornell and Bjork (2008)

Same class of objects?





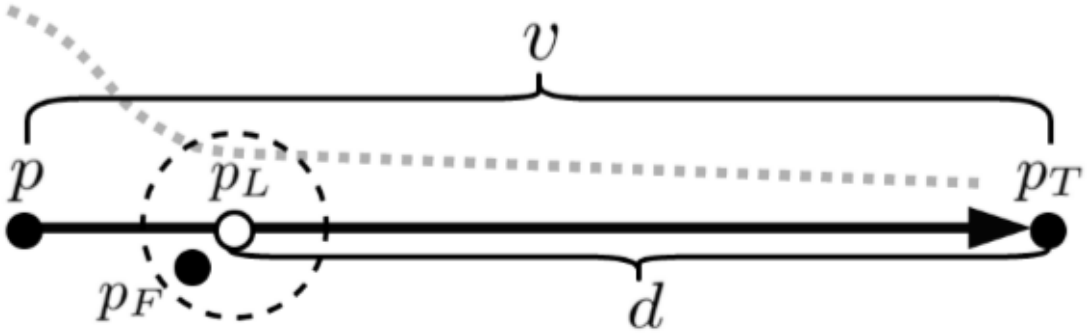

Interleaving Learning for Optical Flow

Controlling the negative sample to balance difficulty

Original method

Interleaving

Interleaving Learning for Optical Flow

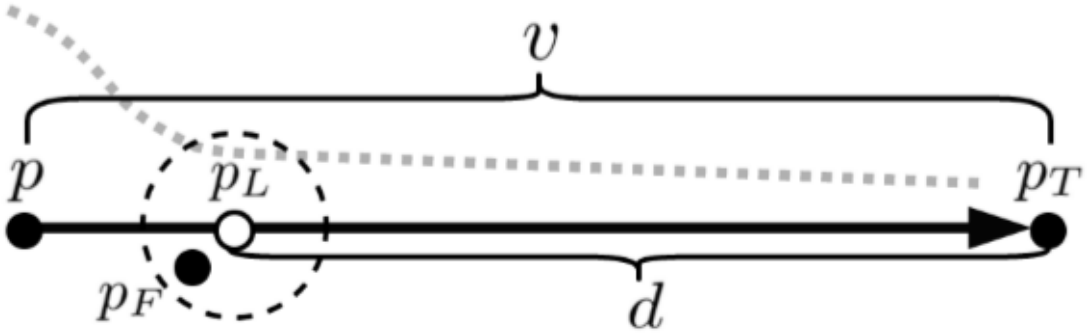


$$d = v(1 - X) \qquad X \sim \log \mathcal{N}(\mu, \sigma)$$

$$P(X = x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{(-\frac{(ln(x)-\mu)^2}{2\sigma^2})}$$

- Drawing the line from p improved matching results but did not effect the distractors measurement (Due to PatchMatch initialization)

Self-Paced Curriculum Interleaving Learning (SPCI)



$$d = v(1 - X) \qquad X \sim \log \mathcal{N}(\mu, \sigma)$$

$$P(X = x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{(-\frac{(ln(x)-\mu)^2}{2\sigma^2})}$$

$$d_i = v(1 - X - R_i)$$

$$R_i = \underbrace{\frac{i}{m}}_{\text{curriculum}} \cdot \underbrace{\max(0, 1 - \frac{l_{i-1}}{l_{init}})}_{\text{self-paced}}$$

l_i - validation loss on epoch i

l_{init} - initial loss value (epoch #5)

m – total epoch amount

Experiments

Optical Flow

Model / Learning method	Error percent		Distractors amount by displacement range								
	post PM	post EF	0-5	5-10	10-20	20-30	30-45	45-60	60-90	90-∞	All
CENT [3]	9.93%	5.19%	3.31	15.34	16.87	27.61	48.28	69.19	92.62	209.13	32.86
CENT+SD [3]	8.91%	4.85%	4.33	16.7	12.29	19.92	38.20	60.69	81.22	216.02	28.67
CENT+SD / **Inter**	8.75%	4.70%	2.61	10.50	8.64	15.29	30.38	42.87	66.16	137.81	20.73
Hinge	7.78%	5.18%	1.93	8.14	5.81	10.98	31.95	50.97	73.24	185.81	21.40
Hinge+SD	7.74%	4.85%	2.32	7.32	5.32	9.38	25.21	50.43	67.32	216.39	20.51
Hinge+SD / Neg-mining	7.53%	5.00%	3.06	6.19	5.41	10.52	26.88	51.33	70.29	210.34	20.96
Hinge+SD / Cur. by disp	7.67%	4.83%	2.71	8.61	5.26	10.26	14.76	48.88	65.15	220.13	20.67
Hinge+SD / Cur. by dist	7.47%	4.93%	2.83	8.66	5.25	10.35	23.62	45.82	63.69	197.82	19.70
Hinge+SD / Self-Paced	8.75%	5.23%	2.88	9.35	6.84	13.74	34.09	57.46	80.8	198.97	23.93
Hinge+SD / Anti-Inter	14.53%	8.30%	2.98	9.12	13.36	20.63	37.69	42.41	81.41	132.03	24.11
Hinge+SD / **Inter**	6.60%	4.41%	1.41	5.57	3.07	6.31	15.6	28.52	43.46	127.65	12.61
Hinge+SD / **SPCI**	6.64%	4.37%	1.40	5.04	3.46	6.56	15.11	27.13	42.72	130.17	12.50
Hinge+SD+PS71	7.34%	4.76%	1.96	5.44	5.28	11.8	22.76	42.3	67.27	190.3	18.91
Hinge+SD+PS71 / **Inter**	6.17%	4.35%	**1.00**	3.96	2.22	4.11	11.33	**20.87**	32.53	**119.74**	9.80
Hinge+SD+PS71 / **SPCI**	**6.12%**	**4.27%**	1.02	**3.42**	**2.16**	**3.52**	**10.55**	21.28	**32.17**	119.98	**9.54**

Results

Benchmarks - KITTI2012

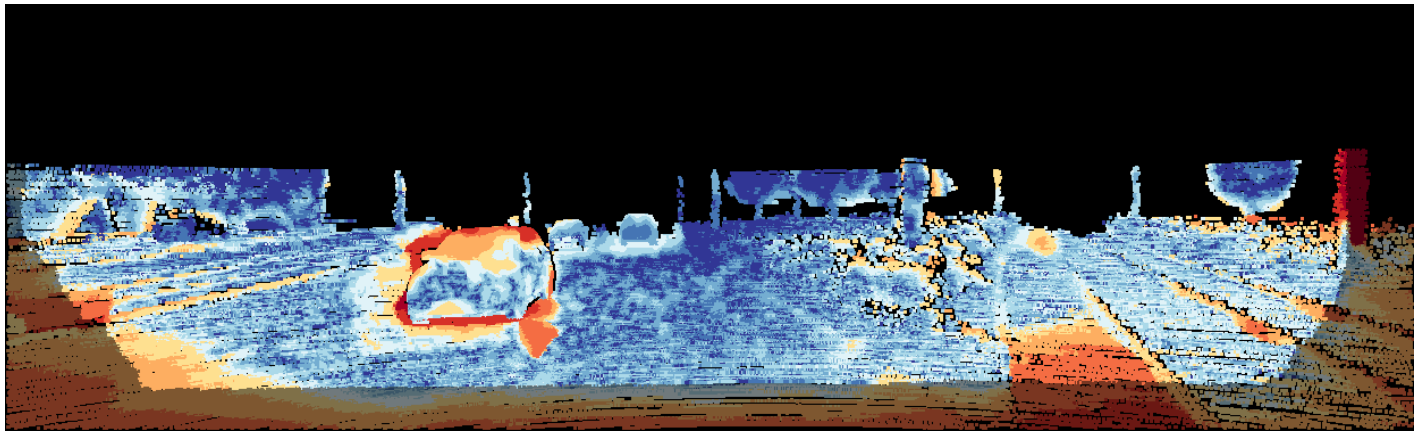
Method	Out-Noc
Imp. PatchBatch+SPCI	**4.65%**
CNN-HPM [20]	4.89%
Imp. PatchBatch	4.92%
PatchBatch+PS71 [3]	5.29%
PatchBatch [3]	5.44%
PH-Flow [34]	5.76%
FlowFields [35]	5.77%
CPM-Flow [36]	5.79%

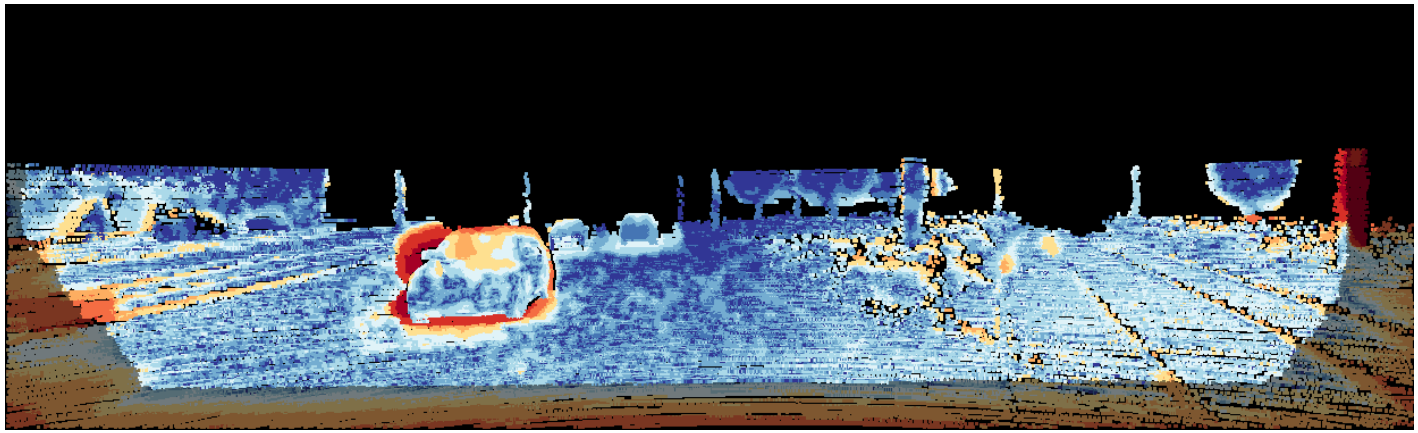
Table 6.4: Top 8 published KITTI2012 Pure Optical Flow methods as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in chapter 3. Out-Noc is the percentage of pixels with euclidean error > 3 pixels out of the non-occluded pixels.

Benchmarks - KITTI2015

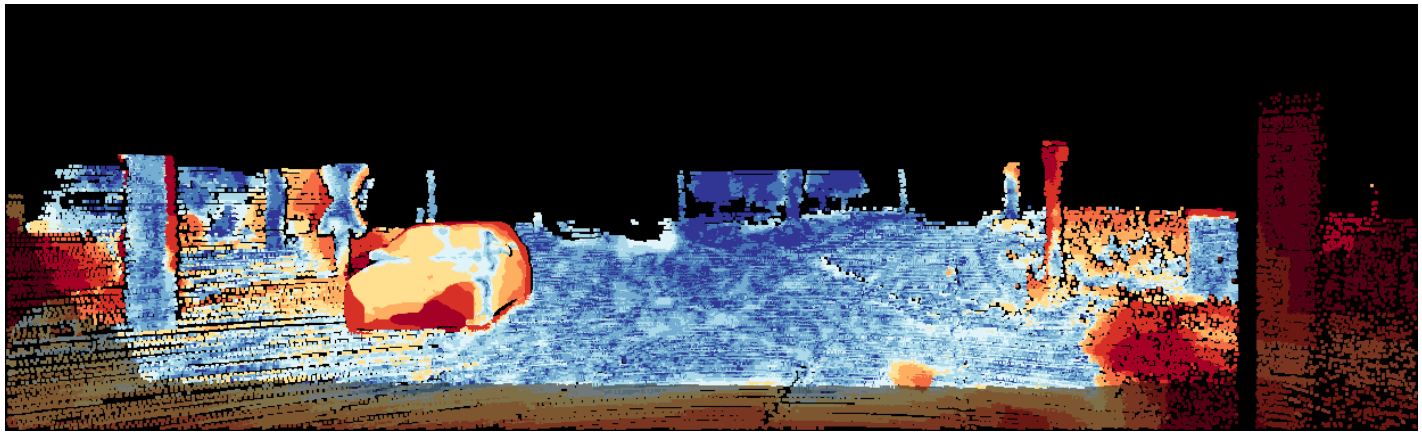
Method	Fl-bg	Fl-fg	Fl-all
Imp. PatchBatch+SPCI	**17.25%**	**24.52%**	**18.46%**
CNN-HPM [20]	18.90%	24.96%	19.44%
PatchBatch [3]	19.98%	30.24%	21.69%
DiscreteFlow [10]	21.53%	26.68%	22.38%
CPM-Flow [36]	22.32%	27.79%	23.23%
FullFlow [37]	23.09%	30.11%	24.26%
EpicFlow [27]	25.81%	33.56%	27.10%
DeepFlow [9]	27.96%	35.28%	29.18%

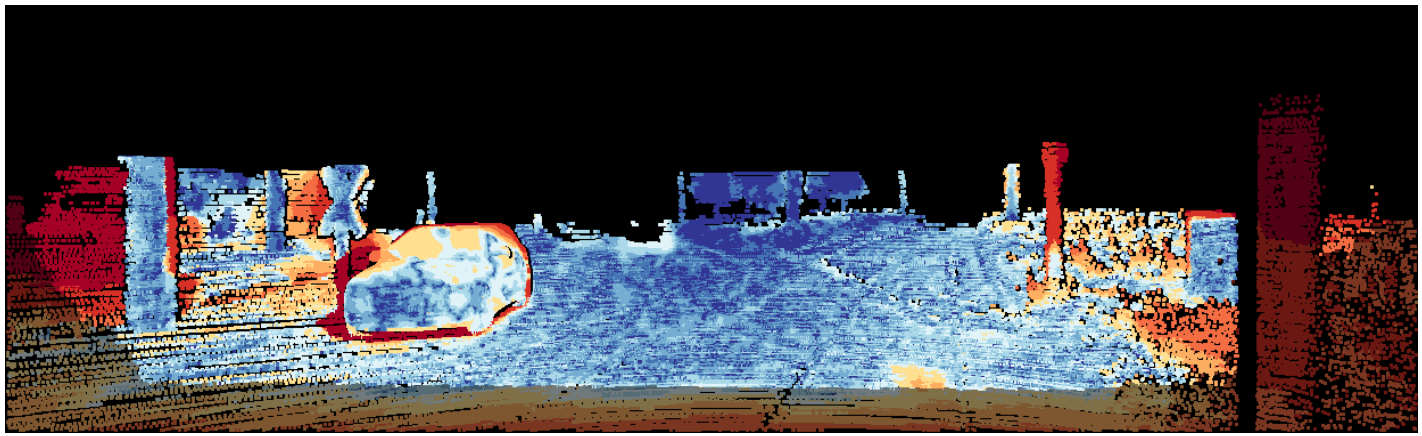
Table 6.5: Top 8 published KITTI2015 Pure Optical Flow methods as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in chapter 3. Fl-all is the percentage of outliers (pixels with euclidean error > 3 pixels). Fl-bg, Fl-fg are the percentage of outliers only over background and foreground regions respectively.

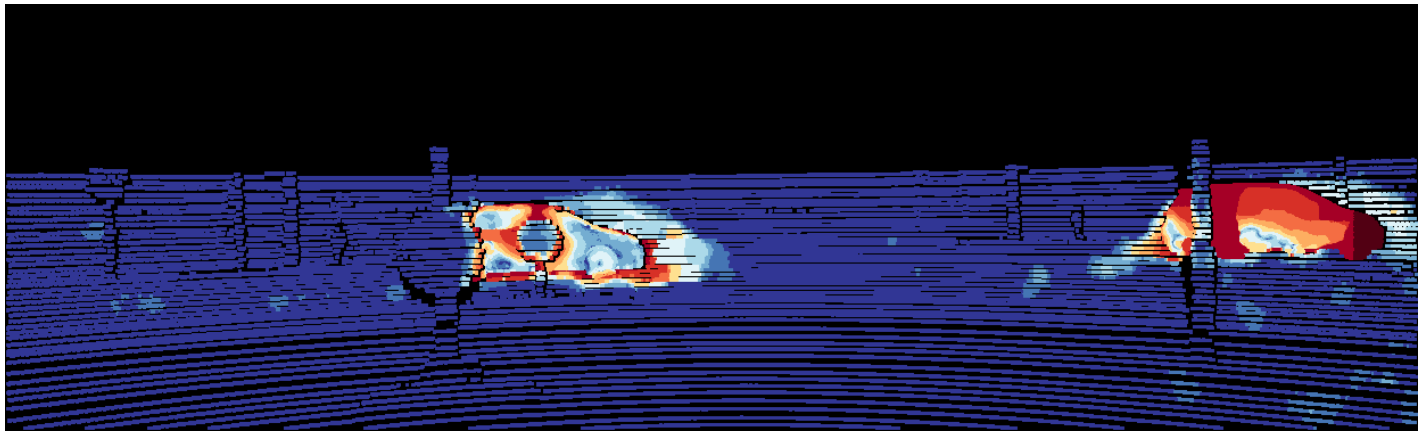


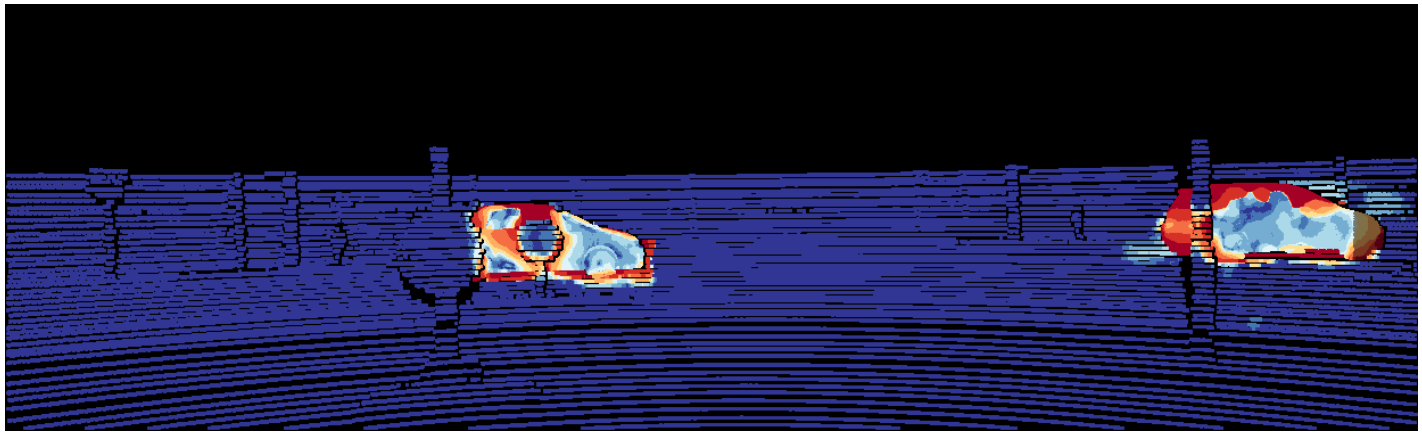



V









Benchmarks - MPI-Sintel

Method	EPE	Fl	s0-10	s40+
FlowFields+ [35]	**5.71**	8.14%	1.31	**34.17**
DeepDiscreteFlow [38]	5.73	**7.30%**	0.96	35.82
SPM-BPv2 [39]	5.81	9.17%	1.05	35.12
FullFlow [37]	5.90	9.55%	1.14	35.59
CPM-Flow [36]	5.96	8.31%	1.15	35.14
GlobalPatchCollider [40]	6.04	10.21%	1.10	36.45
DiscreteFlow [10]	6.08	9.52%	1.07	36.34
Imp. PatchBatch+Inter	6.22	8.11%	0.91	39.91
Imp. PatchBatch+SPCI	6.24	7.89%	0.88	40.07
EpicFlow [27]	6.28	11.26%	1.13	38.02
FGI [41]	6.61	12.34%	1.15	39.98
TF+OFM [11]	6.73	11.35%	1.51	39.76
Deep+R [42]	6.77	13.71%	1.16	41.69
PatchBatch [3]	6.78	8.66%	**0.72**	45.86

Table 6.6: Comparison of our models with the top methods for the MPI-Sintel benchmark as of the submission date. Imp. PatchBatch denotes the PB pipeline with the improvements described in chapter 3. The EPE (end-point-error) is averaged over all the pixels and the two right columns contain only the EPE of pixels within the displacement range mentioned in the title. The Fl column presents an evaluation of the the outlier percentage, which, although not provided by this benchmark, was calculated from the error figures presented for each scene that have higher pixel values for larger errors. Fl is the percentage of pixels with a value larger than 120.

Summary

Summary

- Computing Optical Flow as a matching problem with a modular pipeline
- using a CNN to generate descriptors
- Per-batch statistics (SD, batch normalization)
- Interleaving Learning Method & SPCI
  Referring difficulties while maintaining a random order of the categories
- One model to generate descriptors for both small and large displacements

THANK YOU!