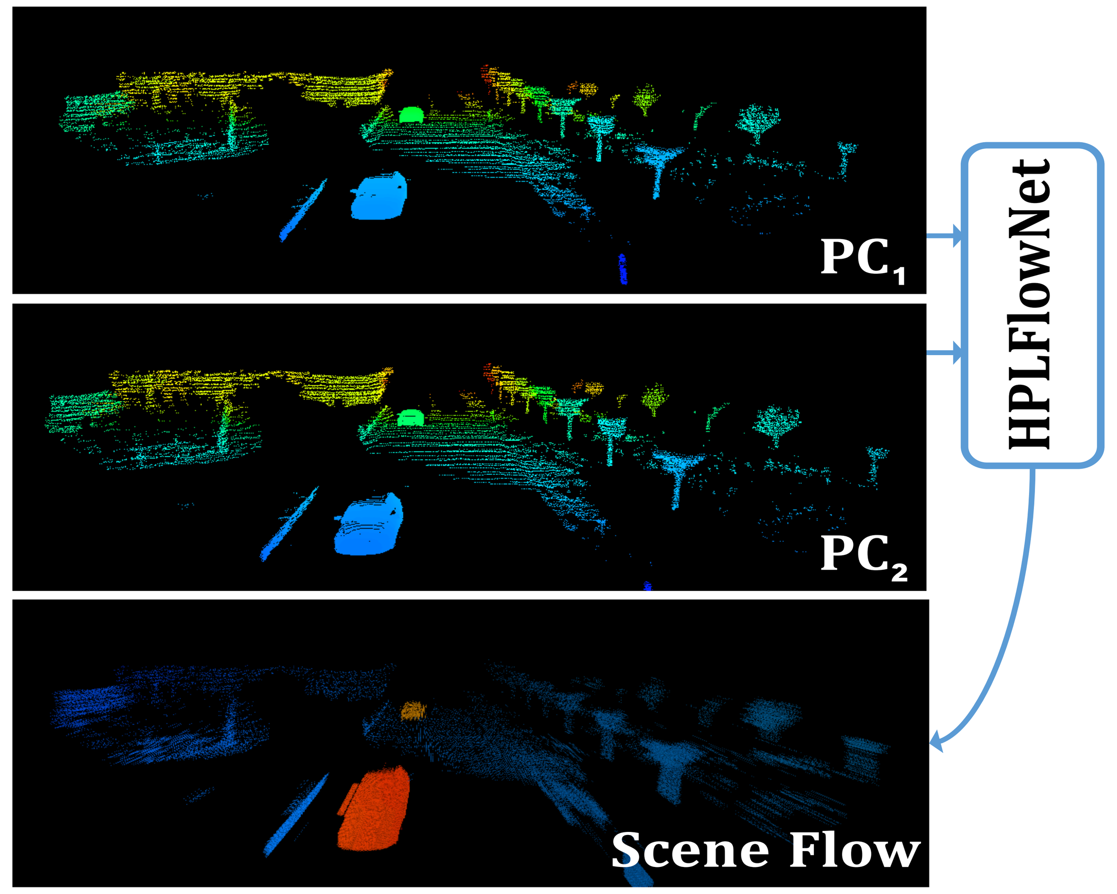
HPLFlowNet: *H*ierarchical *P*ermutohedral *L*attice FlowNet for Scene Flow Estimation on Large-scale Point Clouds

Xiuye Gu[1,2], Yijie Wang[3], Chongruo Wu[2], Yong Jae Lee[2], Panqu Wang[3]
[1]Stanford, [2]UC Davis, [3]TuSimple

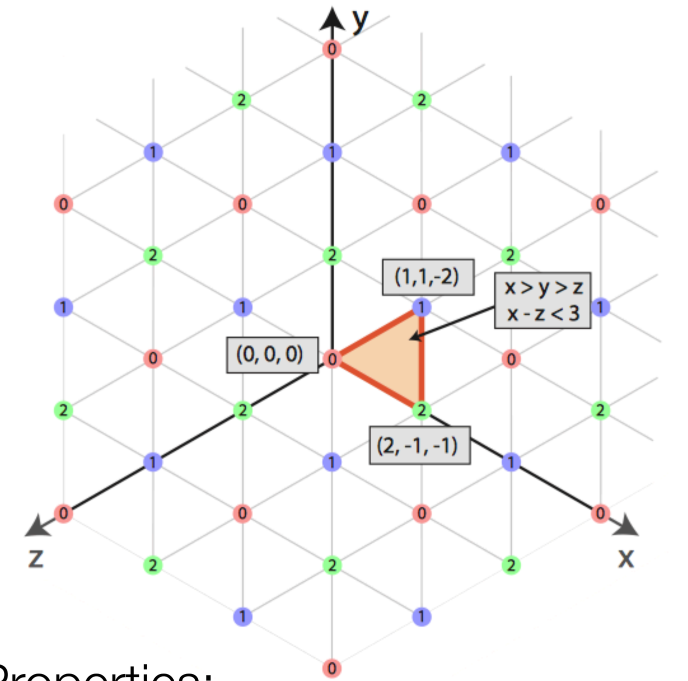
Goal & Background



Goal: *Effectively* and *efficiently* estimate scene flow (3d counterpart of optical flow) for large-scale point clouds.

Q: How can we better restore structural information from unordered point clouds?

-- Interpolate to *permutoheral lattice*[1]
& perform convolution with kernel size > 1 on the lattice.



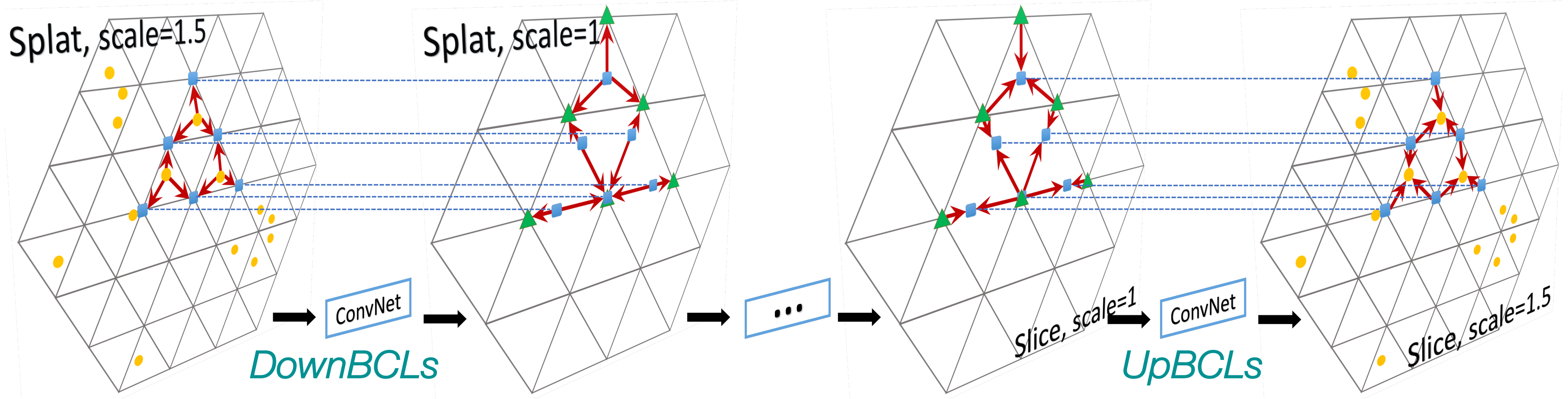
Definition: The projection of the scaled regular grid $(d+1)\mathbb{Z}^{d+1}$ along $\vec{1}$ onto the hyperplane $H_d\colon \vec{x}\cdot\mathbb{1}=0$.

Properties:
1) Lattice tessellates plane with uniform d-simplices.
2) Vertices of simplex containing any point in H_d & nearest neighbors of lattice point can be computed in $O(d^2)$ time.
-- In contrast to $O(2^d)$ in regular grid for both numbers.

[1] A. Adams, J. Baek and M.A. Davis. Fast high-dimensional filtering using the permutohedral lattice.

Approach

Q: How can we process the entire large-scale point cloud at once efficiently?

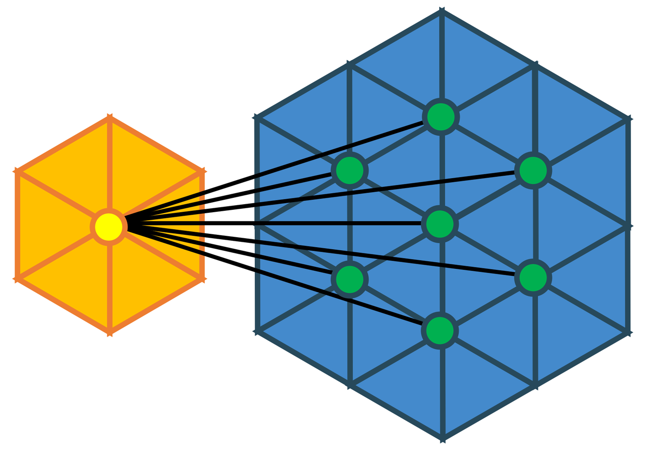


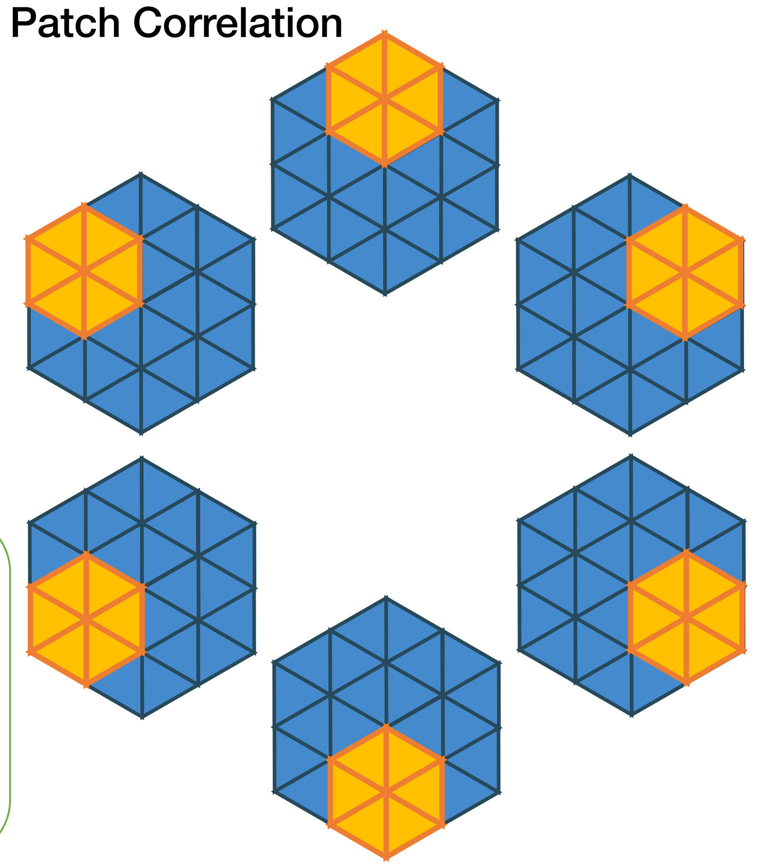
	Original BCL	DownBCL	UpBCL
Pipeline	Splat – Conv – Slice	Splat – Conv	Conv – Slice
# points involved in each operation • G_i: # non-empty lattice points for ith layer • $N > G_1 > G_2 > G_3 > \dots$	$N - G_1 - N$, $N - G_2 - N$, $N - G_3 - N$, …	$N - G_1$, $G_1 - G_2$, $G_2 - G_3$, …	…, $G_3 - G_2$, $G_2 - G_1$, $G_1 - N$

Majority of our network operates on sparse lattice points → computational cost only relates to the actual volume point clouds occupy.

- Sparse conv: Only perform convolution on non-empty lattice points stored in hash tables.
- *Save computational cost w/o sacrificing accuracy as per analysis & experiments.*

Q: How can we better fuse information from the two point clouds? -- *CorrBCLs.*





Implementation

Patch Correlation:
$(C_{in}, p)\oplus(C_{in}, p) \longrightarrow (2C_{in}, p) \xrightarrow{\text{ConvNet}} (C_{corr\_out}, 1)$

Displacement Filtering:
$(C_{corr\_out}, q) \xrightarrow{\text{ConvNet}} (C_{filter\_out}, 1)$

C_{xxx}: # channels of xxx; p/q: neighborhood size (both = 7 in this case)

- Patch correlation
  - ~ matching cost
  - Concatenation + ConvNet
  - Better than non-deep correlation (ablation studies)
- Displacement filtering
  - Avoid brute force
  - Deep aggregation
- Factorization technique
  - # params = $O(p+q)$ not $O(pq)$

Q: How can we deal with different point densities?

-- *Point density normalization* at Splat stage.
Efficient & CorrBCL can also use the normalization scheme.

$$u_j = \frac{\sum_{k\in\mathcal{V}(j)} b_{kj}\cdot v_k}{\sum_{k\in\mathcal{V}(j)} b_{kj}}$$

b_{kj}: barycentric interpolation weight
v_k: signal values

Quantitative Results

Comparison & Generalization (on KITTI)

-- Our method outperforms all baseline methods.

Dataset	Method	EPE3D (cm)↓	Acc Strict↑	Acc Relax↑	Outliers↓
FlyingThings3D	FlowNet3	45.70	41.8%	61.7%	60.5%
	ICP	40.62	16.1%	30.4%	88.0%
	FlowNet3D	11.36	41.3%	77.1%	60.2%
	SPLATFlowNet	12.05	42.0%	71.8%	61.9%
	Original BCL	11.11	42.8%	75.5%	60.5%
	Ours	8.04	61.4%	85.6%	42.9%
KITTI	FlowNet3	91.11	20.4%	35.9%	74.6%
	ICP	51.81	6.7%	16.7%	87.1%
	FlowNet3D	17.67	37.4%	66.8%	52.7%
	SPLATFlowNet	19.88	21.7%	53.9%	65.8%
	Original BCL	17.29	25.2%	60.1%	62.2%
	Ours	11.69	47.8%	77.8%	41.0%

Memory efficiency

-- Can process a pair of point cloud frames at once with a maximum of 86K points per frame.

EPE3D under different point densities

-- Our density normalization scheme works well.

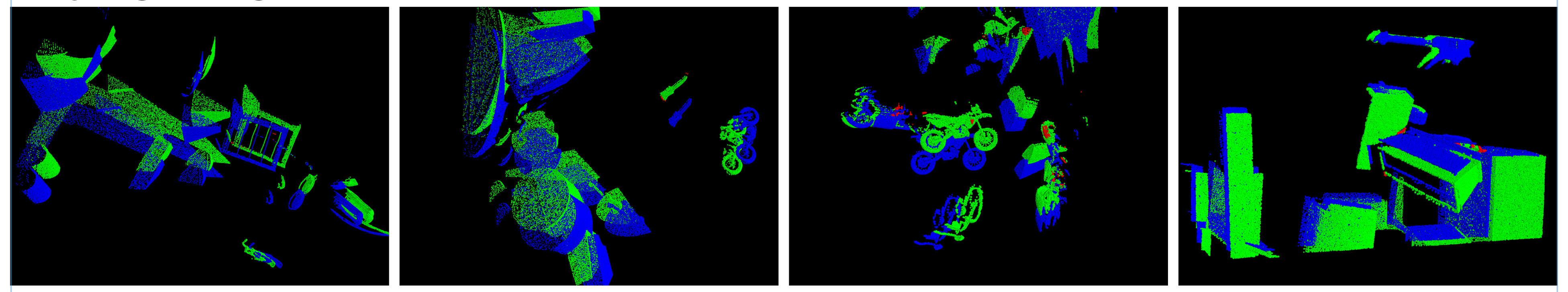
Dataset	# points	FlowNet3D	No Norm	Ours-shallow	Ours
FlyingThings3D	8,192	11.36	7.90	9.57	8.04
	16,394	10.85	7.79	9.32	7.82
	32,768	13.27	8.74	9.25	7.74
	65,536	-	12.67	9.25	7.72
KITTI	8,192	17.67	11.87	16.30	11.69
	16,384	20.95	13.05	16.46	11.14
	32,768	31.10	16.63	16.71	10.87
	65,536	-	18.42	16.74	10.87
	All points	-	18.53	16.74	10.87

Time efficiency

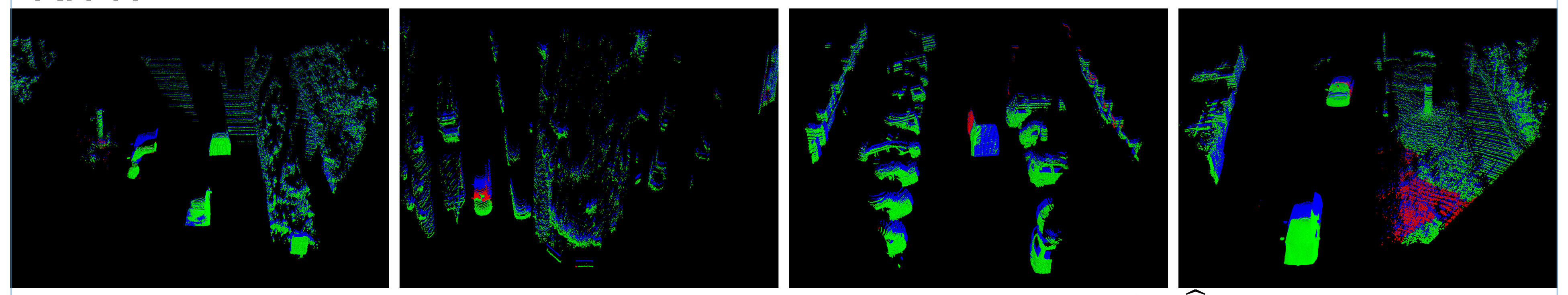
-- Average runtime 98.4/115.5/142.8/193.2ms for 8,192/16,384/32,768/65,536 points on FlyingThings3D using a single Titan V.

Qualitative Results

FlyingThings3D



KITTI



Blue points are PC$_1$, green points are correctly predicted flowed points PC$_1$+$\widehat{sf}$, and red points are ground-truth flowed points PC$_1$+sf which are not correctly predicted.

Code released at https://github.com/laoreja/HPLFlowNet