Video Propagation Networks

V. Jampani, R. Gadde and P. V. Gehler, CVPR 2017

Jonáš Šerých

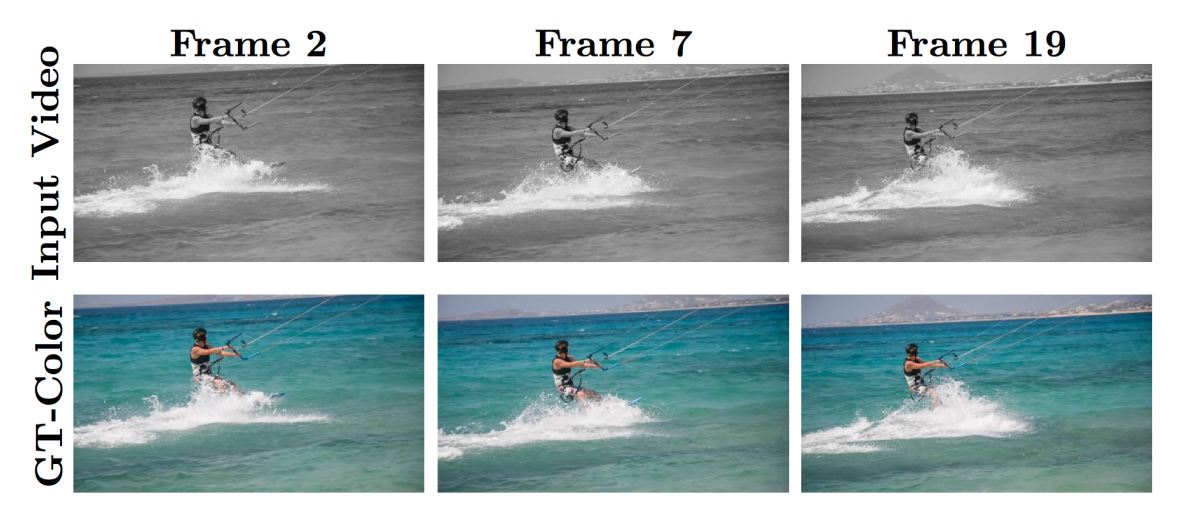
2019-09-05

The Task

Given:

- Video sequence
- Per-pixel information (color, segmentation, . . .) on few frames

Propagate the information to the whole video.

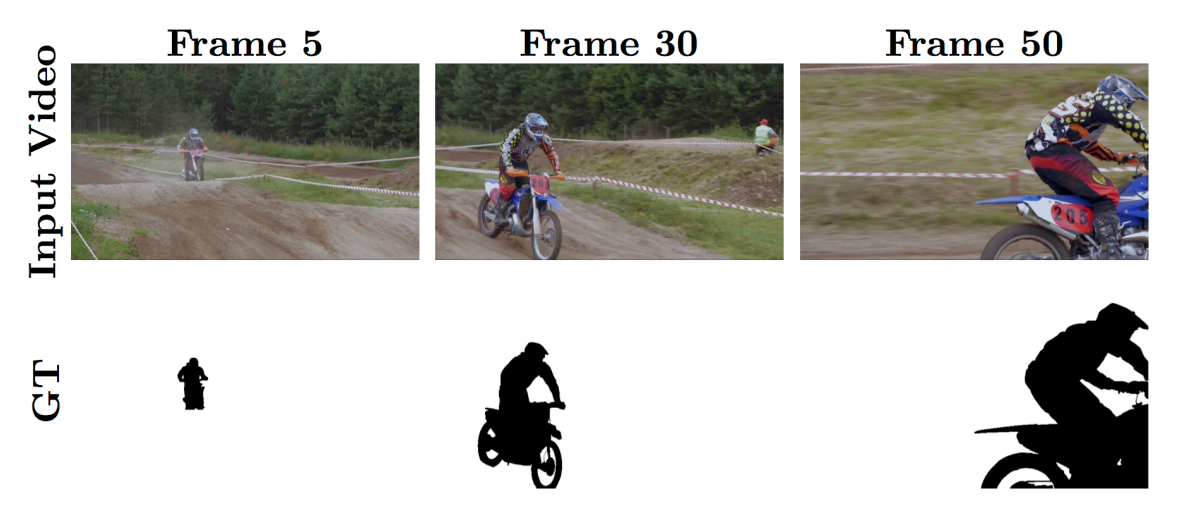


The Task

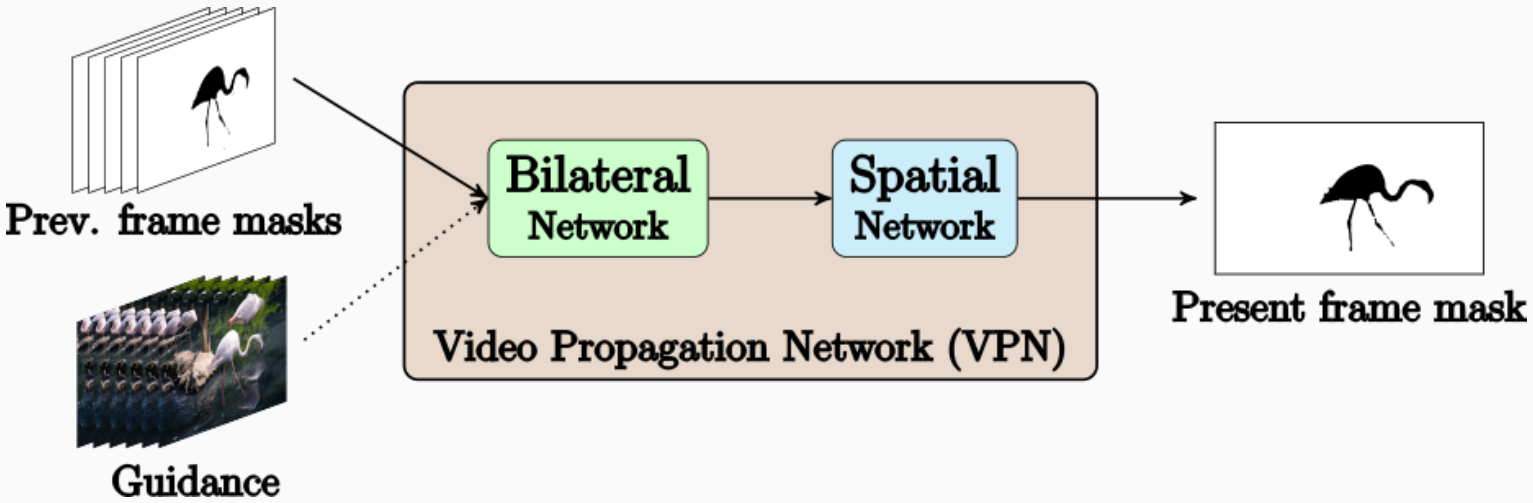
Given:

- Video sequence
- Per-pixel information (color, segmentation, . . .) on few frames

Propagate the information to the whole video.



The Approach



Bilateral network

- image-adaptive spatio-temporal dense filtering
- straight-forward integration of temporal information

Spatial Network

- shallow CNN
- spatial refinement

Bilateral Filtering – Introduction

Standard Gaussian filtering – weighted average of all pixel values:

$$\mathbf{v}'_i \approx \sum_{j=0}^{n} e^{-||\mathbf{p}_i - \mathbf{p}_j||^2} \mathbf{v}_j$$

$$\mathbf{p}_i = (x_i, y_i)$$

- spatially close $\rightarrow$ bigger influence

Bilateral Filtering – Introduction

Standard Gaussian filtering – weighted average of all pixel values:

$$\mathbf{v}'_i \approx \sum_{j=0}^{n} e^{-||\mathbf{p}_i - \mathbf{p}_j||^2} \mathbf{v}_j$$

$$\mathbf{p}_i = (x_i, y_i)$$

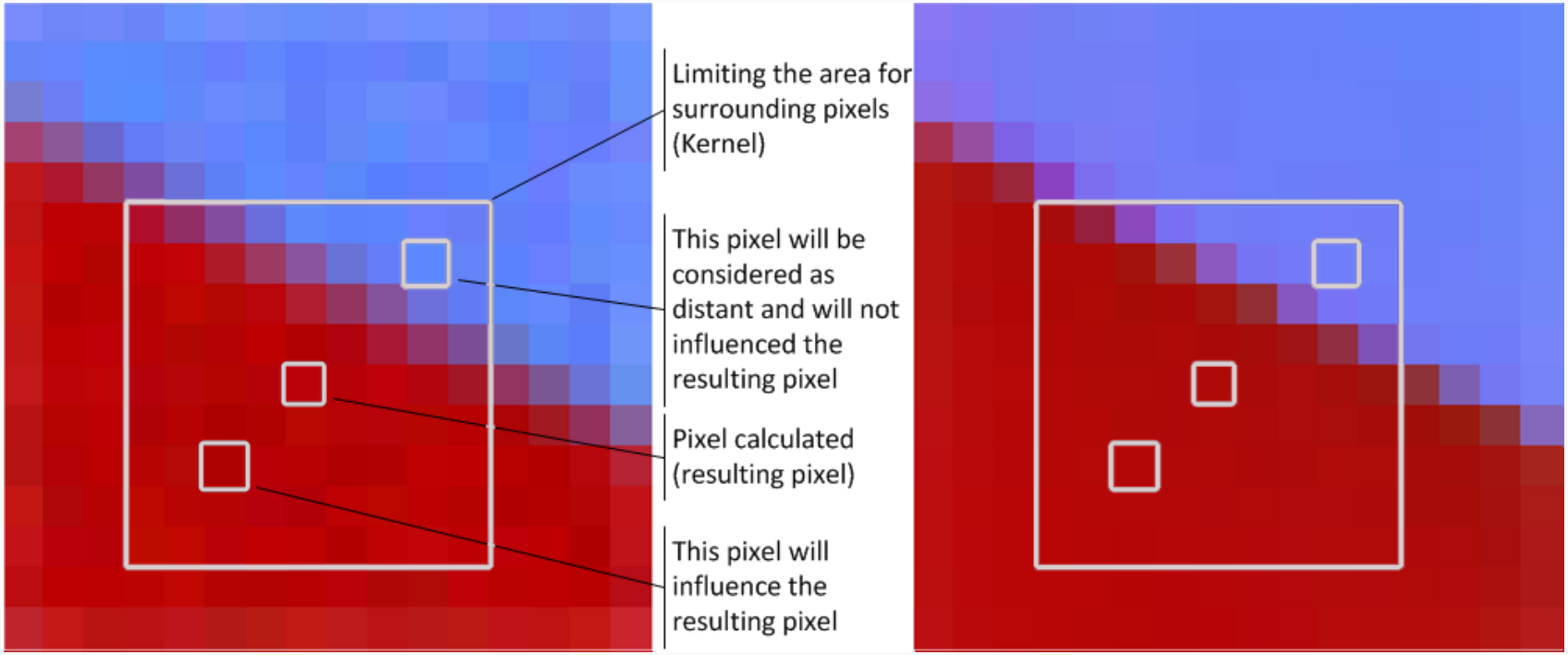
- spatially close $\rightarrow$ bigger influence

Bilateral filtering:

$$\mathbf{p}_i = (x_i, y_i, R_i, G_i, B_i)$$

- spatially close *and visually similar* $\rightarrow$ bigger influence

Edge-Preserving Bilateral Filtering Illustration



https://saplin.blogspot.com/2012/01/bilateral-image-filter-edge-preserving.html

Joint Bilateral Upsampling Illustration

Signal (coloring) on low-resolution image upsampled using high-resolution image guide.

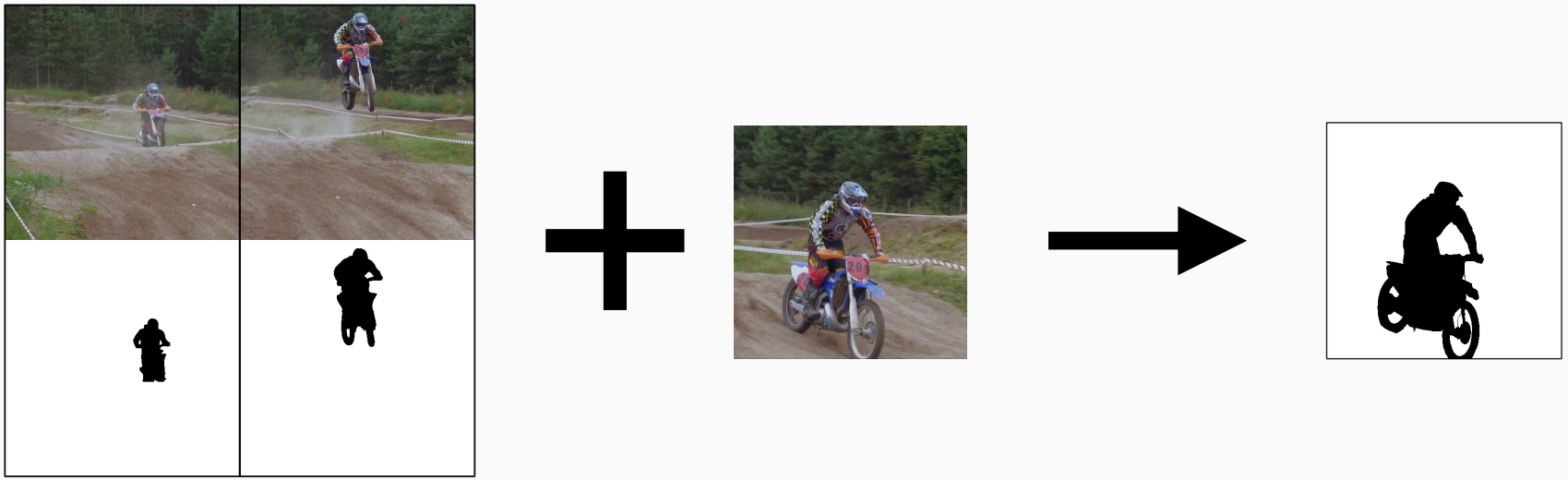
Image from slides by Peter Gehler

Kopf, Johannes, et al. "Joint bilateral upsampling." ACM Transactions on Graphics, 2007.

Bilateral Filtering – Propagation in Video

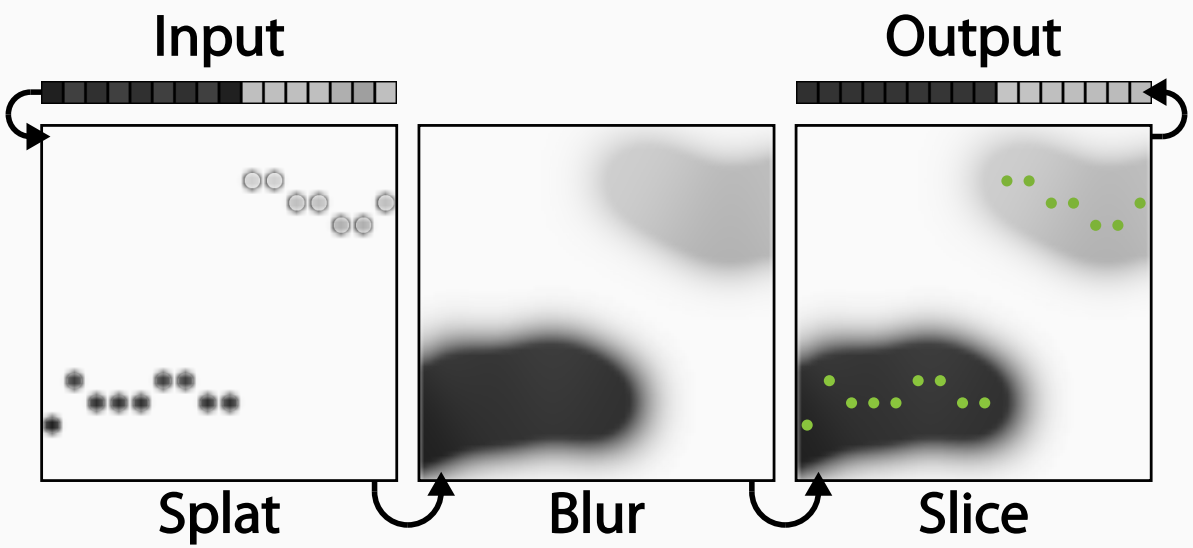
The main idea: Use the current frame as a guide for information propagation from the past frames.

Use (x, y, R, G, B, t) instead of (x, y, R, G, B).



Bilateral Filtering - Implementation Overview

1. **Splat**: Embed input values $\mathbf{v}_i$ at positions $\mathbf{p}_i$ in a high-dimensional space.
2. **Blur**: Perform the filtering.
3. **Slice**: Sample the space at positions $\mathbf{p}'_i$.

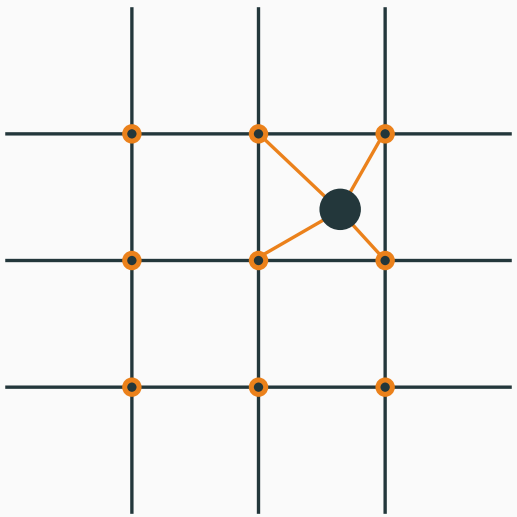


Naive Implementation

2D example:

- Just do a convolution with Gaussian filter.
- But what if the positions are not on the grid?

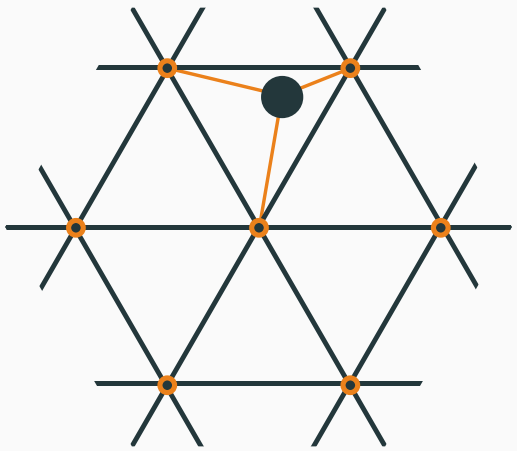
We could *splat* values onto the grid using bilinear interpolation:



OK, **but**: Regular square grid: 2^D neighboring vertices!

Efficient Implementation Using Permutohedral Lattice

Permutohedral lattice: only $D + 1$ neighboring vertices



1. Find the nearest lattice vertices and the corresponding weights in $\mathcal{O}\left(D^2\right)$.
2. Accumulate weighted values in lattice vertices (**splat**).
3. Perform convolution on the lattice (**blur**).
4. Interpolate from the lattice (**slice**).

Linearity of Bilateral Filtering

Given (1-D for simplicity) values $\mathbf{v} \in \mathbb{R}^N$ at positions $\mathbf{p} \in \mathbb{R}^{N \times D}$:

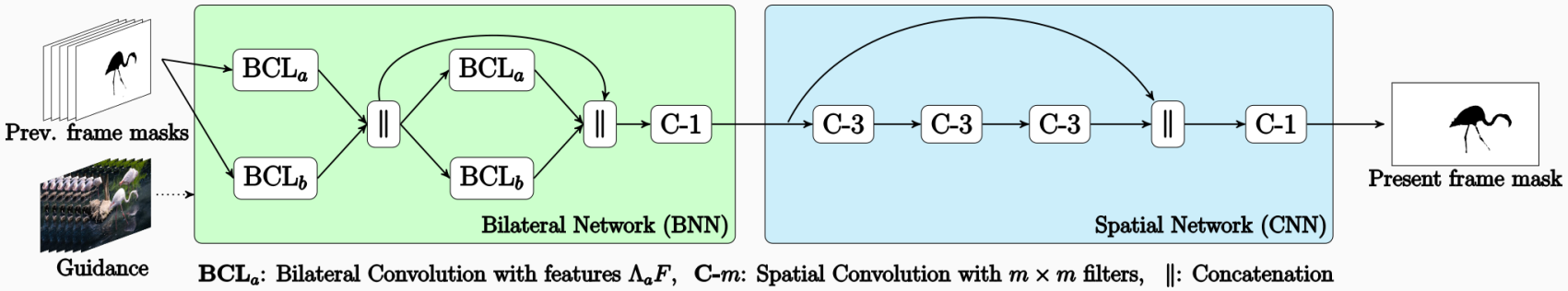
- Construct $S_{splat} \in \mathbb{R}^{M \times N}$ using $\mathbf{p}$.
  M ... number of lattice points.
  Each column of S_{splat} contains the weights of single input.
- Construct convolution in the matrix form $B \in \mathbb{R}^{M \times M}$.
- Construct $S_{slice} \in \mathbb{R}^{N \times M}$ similarly to S_{splat}.

Then: $\mathbf{v}' = S_{slice}\left(B\left(S_{splat}\mathbf{v}\right)\right)$

Linear in $\mathbf{v}$ and the convolution weights inside B.

Backpropagation possible.

VPN Architecture



- splat in the first $BCL_{a,b}$ layers guided by previous frames
- the rest guided by the current frame
- ReLU after concatenations and spatial convolutions
- $\Lambda_{a,b}$ position scales found by validation

Some Setup Details

- splice: random sampling or superpixels (12000)
- bilateral convolutions with no neighborhood
- YCbCr instead of RGB
- weighting previous 9 frame values by $\alpha, \alpha^2, \alpha^3, \ldots$, where $\alpha = 0.5$ (!!!)
- optical flow for transformation of positions into current frame
- multi-stage training and inference

Object Segmentation Results

	IoU↑	$\mathcal{F}$ ↑	$\mathcal{T}$ ↓	*Runtime*(s)
BNN-Identity	67.0	67.1	36.3	0.21
VPN-Stage1	70.1	68.4	30.1	0.48
VPN-Stage2	71.3	68.9	30.2	0.75
With pre-trained models				
DeepLab	57.0	49.9	47.8	0.15
VPN-DeepLab	**75.0**	**72.4**	29.5	0.63
OFL [75]	71.1	67.9	22.1	>60
BVS [53]	66.5	65.6	31.6	0.37
NLC [25]	64.1	59.3	35.6	20
FCP [60]	63.1	54.6	28.5	12
JMP [26]	60.7	58.6	**13.2**	12
HVS [29]	59.6	57.6	29.7	5
SEA [62]	55.6	53.3	13.7	6

Semantic Segmentation Results

	IoU	*Runtime*(s)
CNN-1 from [79]	65.3	0.38
+ FSO-CRF [43]	66.1	>10
+ BNN-Identity	65.3	0.31
+ BNN-Identity-Flow	65.5	0.33
+ VPN (Ours)	66.5	0.35
+ VPN-Flow (Ours)	**66.7**	0.37
CNN-2 from [65]	68.9	0.30
+ VPN-Flow (Ours)	**69.5**	0.38

Color Propagation Example Outputs

Conclusions

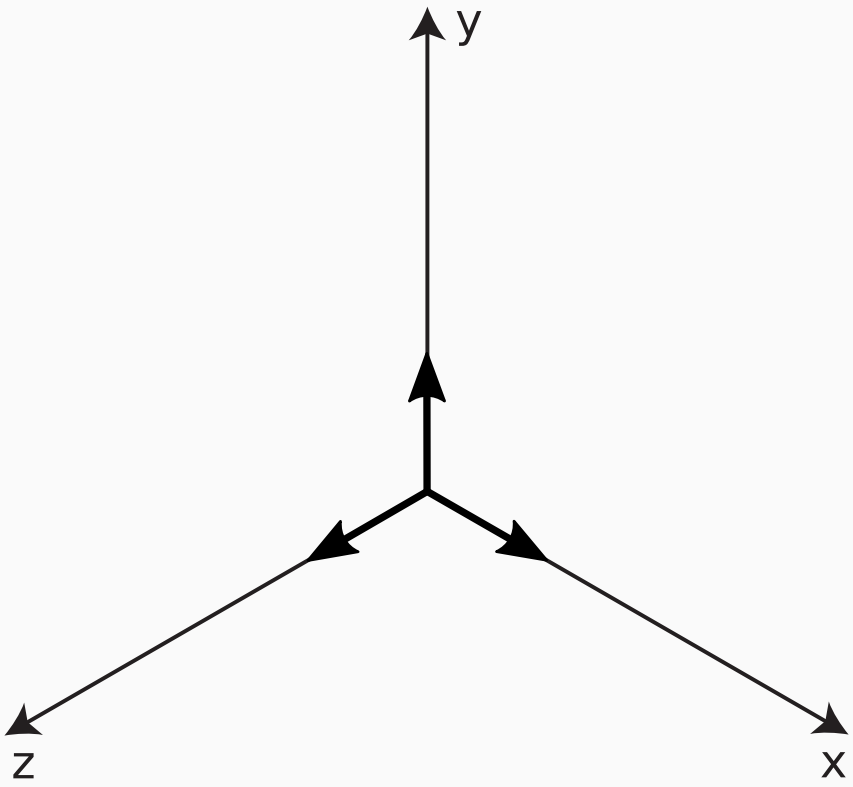
- Efficient implementation of high-dimensional convolutions using permutohedral lattices
- Fast propagation of arbitrary data in video sequences

- Interested? Check: H. Su, V. Jampani et al. Pixel-Adaptive Convolutional Neural Networks (CVPR2019)

Splat with Permutohedral Lattice

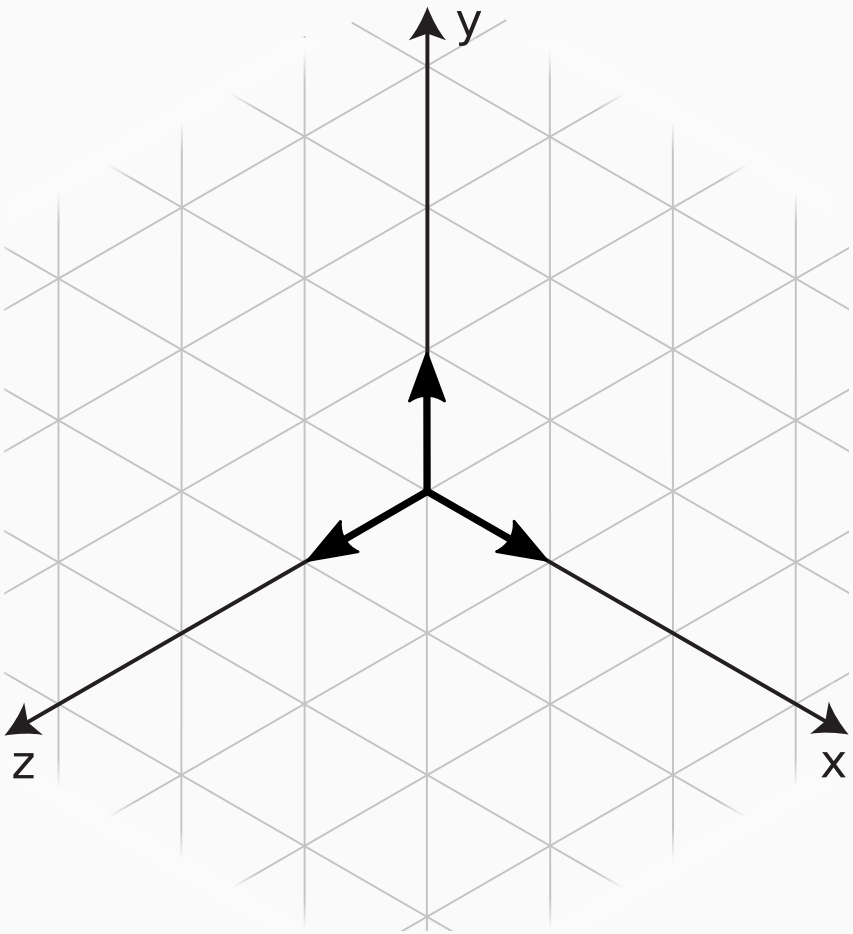
1. Take the hyperplane of $\mathbb{R}^{D+1}$ in which coordinates sum to zero. $H_D : \mathbf{x} \cdot \mathbf{1} = 0$
2. The hyperplane H_D is spanned by "base" vectors: $(D, -1, \ldots, -1)$, $(-1, D, -1, \ldots, -1)$, $\ldots$, $(-1, \ldots, -1, D)$
3. Integer combinations of the "base" vectors are the lattice vertices.

Splat with Permutohedral Lattice



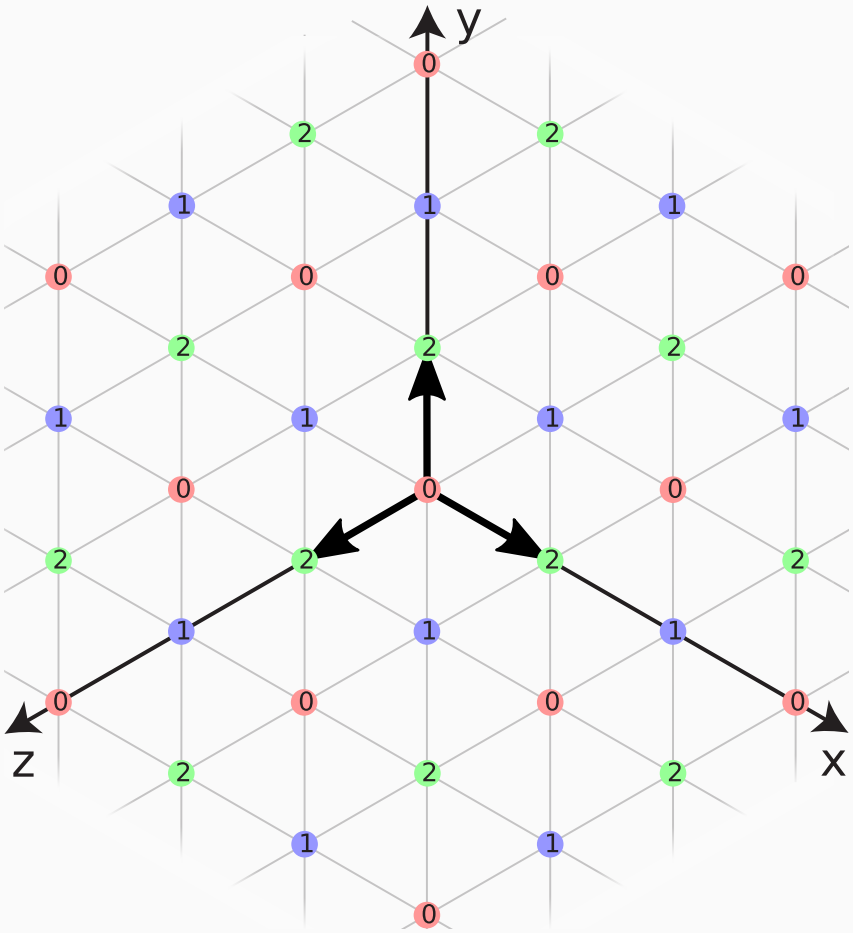
View orthogonal to the hyperplane.

Splat with Permutohedral Lattice



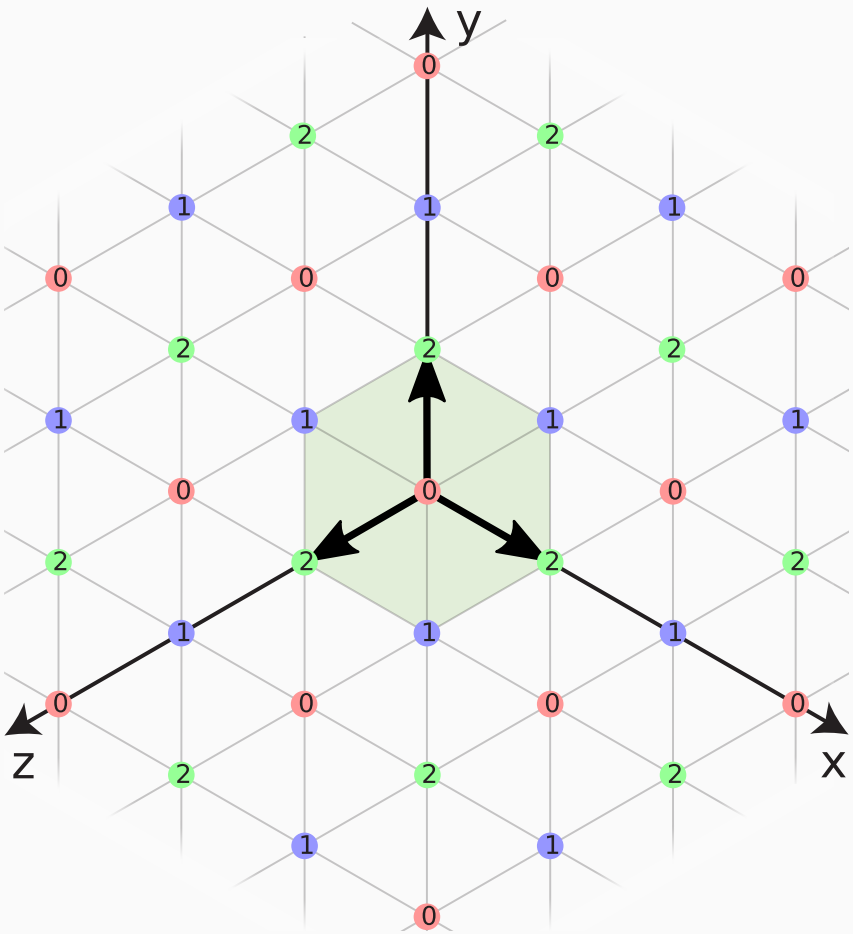
Integer combinations of the “base” vectors form the lattice.

Splat with Permutohedral Lattice



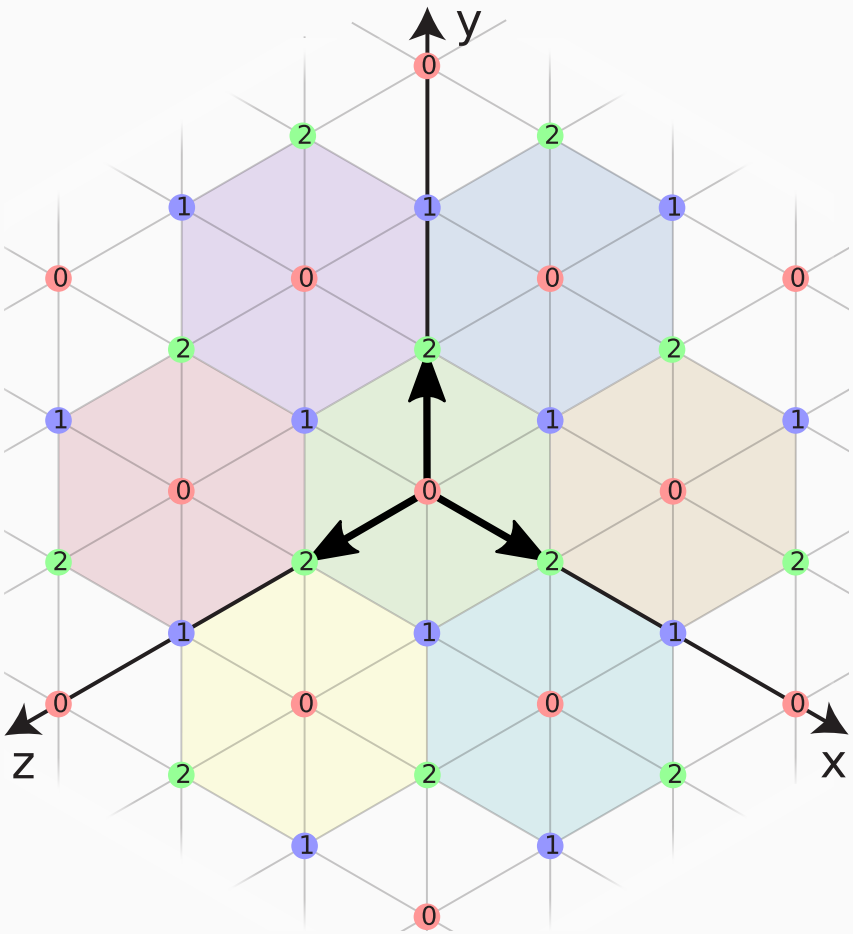
Each vertex has consistent coordinates modulo $(D + 1)$.

Splat with Permutohedral Lattice



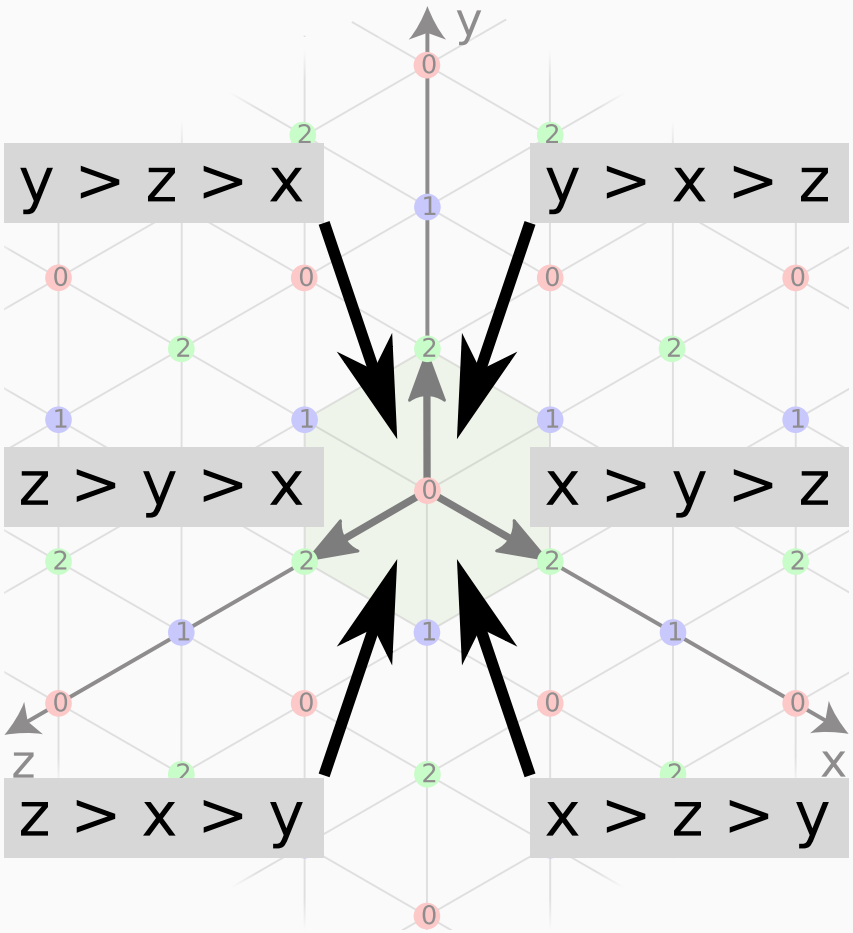
Permutohedron formed by the lattice points.

Splat with Permutohedral Lattice



The H_D hyperplane is tiled by translations of the permutohedron.

Splat with Permutohedral Lattice



The neighboring lattice vertices fully identified by closest 0-remainder point $\mathbf{l}_0$ and coordinate ordering of $\mathbf{x} - \mathbf{l}_0$.

Splat with Permutohedral Lattice

Finding closest remainder-0 vertex

1. $\mathbf{l}_0 \leftarrow$ round coordinates of $\mathbf{x}$ to nearest multiple of $(D+1)$
2. Sort the coordinates by the amount of rounding
3. Iterate starting with the most rounded coordinate:
   3.1 If $\mathbf{l}_0$ lies on H_D: finish
   3.2 Round in the opposite direction
   3.3 Go to the next coordinate

Splat with Permutohedral Lattice

1. Project input position $\mathbf{p}$ into the (D+1)-dimensional hyperplane H_D.
2. Find closest remainder-0 point.
3. Find corresponding simplex.
4. Compute barycentric weights $w_i, i \in \{1, 2, \ldots, D+1\}$.
5. Accumulate the input value $\mathbf{v}$ weighted by w_i into the neighboring lattice vertices (entries in a hash-table).