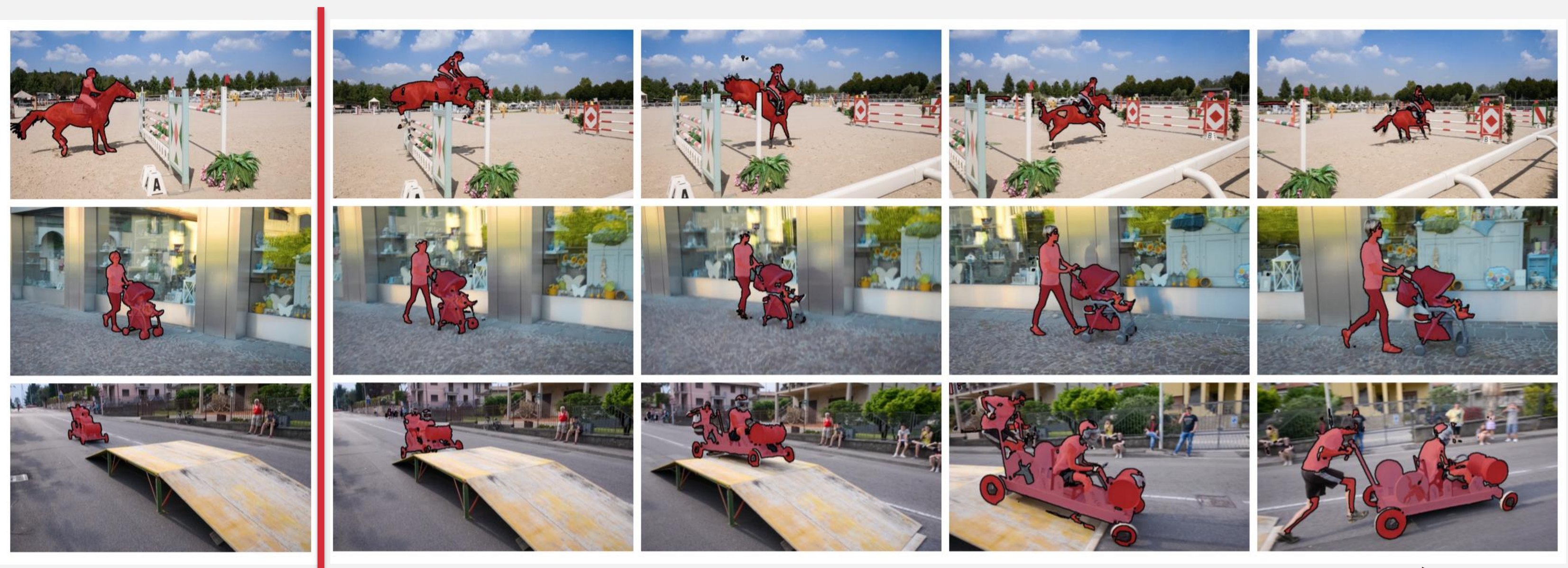
Bilateral Space Video Segmentation

N. Märki[1] F. Perazzi[1,2] O. Wang[3] A. Sorkine-Hornung[2]
[1]ETH Zurich, [2]Disney Research, [3]Adobe Systems Inc.

Introduction



Novel approach to user-driven **video segmentation** in **bilateral space**.

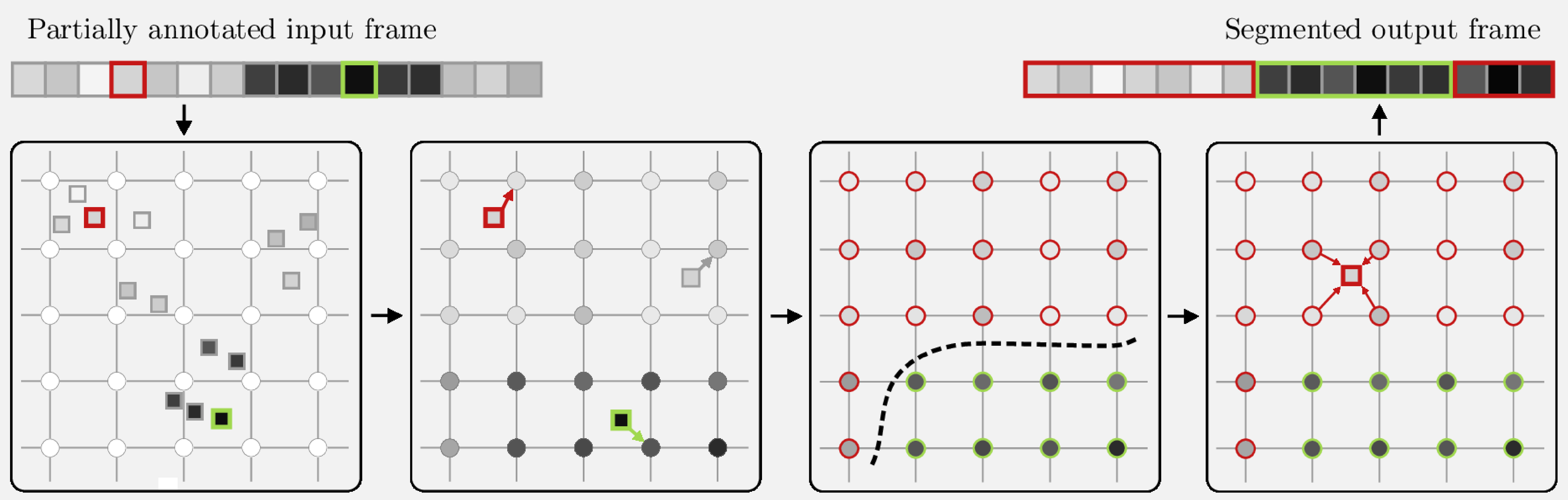
New **energy on the vertices of a spatiotemporal bilateral grid** yields efficient graph cut label assignment.

Energy implicitly approximates long-range, spatio-temporal connections between pixels while still containing only local graph edges.

Visit the project page:
https://graphics.ethz.ch/~perazzif/bvs/index.html
Source Code Available

Method Overview

The algorithm consists of 4 steps: lifting, splatting, graph cut, and slicing.



(a) Lifting (b) Splatting (c) Graph cut (d) Slicing

Lifting

This stage embeds each pixel in a 6D feature space.

We use **YUV** pixel color, **spatial** and **temporal coordinates** as features, which achieves state-of-the-art results while **efficient due to low dimensionality and ease of computation**:

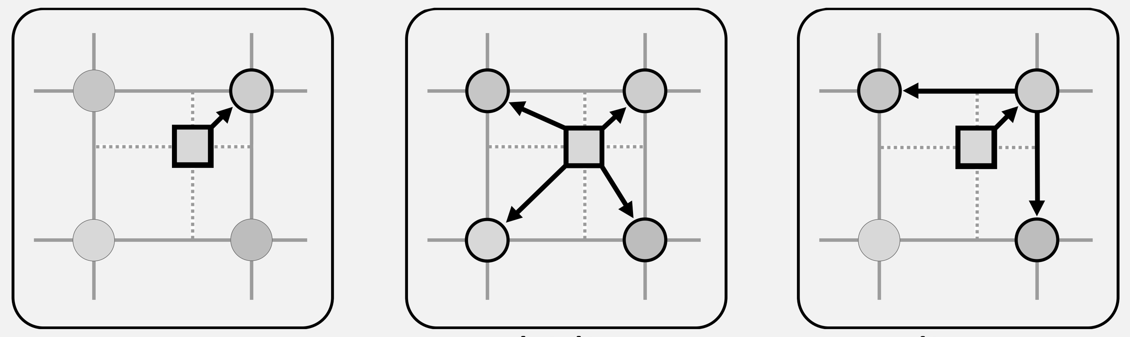
$$\mathbf{b}(\mathbf{p}) = [c_y,\, c_u,\, c_v,\, x,\, y,\, t]^{\mathsf{T}} \in \mathbb{R}^6$$

Splatting

Resampling Instead of labeling each lifted pixel **b(p)** directly, we resample the bilateral space using a regular grid and compute labels on the vertices of this grid **S**. The purpose is to reduce the variables that we will assign labels to in the lifted bilateral space.

Splatting is computed as a weighted sum of samples: $S(\mathbf{v}) = \sum w(\mathbf{v}, \mathbf{b}(\mathbf{p})) \cdot (\hat{\mathbf{p}})$

Interpolation: This weighting function can have different options:



Nearest-neighbor is the fastest, but can lead to blocky artifacts
Multi-linear interpolation is slower, especially with higher dimensional features, but generates higher quality results
Adjacent interpolation provides a good compromise between the two:

Only consider vertices that differ in only one dimension:

$$w_a(\mathbf{v}, \mathbf{b}(\mathbf{p})) = \begin{cases} \prod_{i=1}^{d} |\mathbf{v}_i - N_{\mathbf{b}(\mathbf{p})}| & \text{if } \mathbf{v} \in A_{\mathbf{b}(\mathbf{p})} \\ 0 & \text{otherwise} \end{cases}$$

Graph-Cut

Labeling We now perform a label assignment on the vertices of the bilateral grid. Typically there are many fewer vertices than pixels in the input video, and the connectivity is sparse (neighboring vertices only).

$$E(\boldsymbol{\alpha}) = \sum_{\mathbf{v} \in \Gamma} \theta_{\mathbf{v}}(\mathbf{v}, \alpha_{\mathbf{v}}) + \lambda \sum_{(\mathbf{u},\mathbf{v}) \in \mathcal{E}} \theta_{\mathbf{uv}}(\mathbf{u}, \alpha_{\mathbf{u}}, \mathbf{v}, \alpha_{\mathbf{v}})$$

The data term models deviations from supplied user input (splatted onto the bilateral space vertices). The local connectivity of the smoothness term allows for efficient labeling, while enforcing long range (in pixel-space) constraints.

Slicing

Segmentation **M** is retrieved by slicing the grid labels i.e. interpolating labels at the positions of the lifted pixels in the output frame:

$$\mathcal{M}(\mathbf{p}) = \sum_{\mathbf{v} \in \Gamma} w(\mathbf{v}, \mathbf{b}(\mathbf{p})) \cdot L(\mathbf{v})$$

Evaluation

Our approach is faster (allowing user-interaction) and produces better results.

DAVIS (visit poster 78)

	BVS_Q	BVS_S	JMP	NLC	SEA	HVS
bear	**0.96**	0.93	0.93	0.91	0.91	0.94
blackswan	**0.94**	0.90	0.93	0.87	0.93	0.92
bmx-trees	**0.38**	0.29	0.23	0.21	0.11	0.18
bmx-bumps	0.43	0.41	0.34	**0.63**	0.20	0.43
breakdance-flare	0.73	0.59	0.43	**0.80**	0.13	0.50
breakdance	0.50	0.40	0.48	**0.67**	0.33	0.55
bus	**0.86**	0.84	0.67	0.63	0.75	0.81
dance-twirl	**0.49**	0.35	0.44	0.35	0.12	0.32
libby	**0.78**	0.61	0.29	0.64	0.23	0.55
dog	0.72	0.58	0.67	**0.81**	0.58	0.72
drift-chicane	0.03	0.01	0.24	0.32	0.12	**0.33**
drift-straight	0.40	0.21	**0.62**	0.47	0.51	0.30
mallard-water	**0.91**	0.82	0.75	0.76	0.87	0.70
mallard-fly	0.61	0.61	0.54	**0.62**	0.56	0.44
elephant	**0.85**	0.82	0.75	0.52	0.55	0.74
flamingo	**0.88**	0.72	0.53	0.54	0.58	0.81
goat	0.66	0.58	**0.73**	0.01	0.54	0.58
hike	0.76	0.82	0.66	**0.92**	0.78	0.88
paragliding	0.88	0.84	**0.95**	0.88	0.86	0.91
soccerball	**0.84**	0.57	0.10	0.83	0.65	0.07
surf	0.49	0.62	**0.94**	0.78	0.82	0.76
Average	**0.67**	0.56	0.61	0.64	0.56	0.60

Table 2. IoU score (higher is better) on a representative subset of the *DAVIS* benchmark [26], and the average computed over all 50 sequences.

JumpCut

	BVS_Q	BVS_S	RB	DA	SEA	JMP
animation	**0.78**	1.77	1.98	1.26	1.83	1.59
bball	**1.36**	3.29	1.55	1.71	1.90	1.61
bear	1.34	1.56	1.82	**1.07**	1.84	1.36
car	1.01	5.48	1.35	1.38	0.73	**0.54**
cheetah	**2.72**	3.56	7.17	3.99	5.07	4.41
couple	2.65	6.43	4.09	3.54	3.78	**2.27**
cup	**0.99**	4.54	3.72	1.34	1.19	1.16
dance	**5.19**	23.96	6.65	9.19	7.55	6.62
fish	**1.78**	4.06	2.80	1.97	2.54	1.80
giraffe	4.06	9.89	8.49	6.99	4.77	**3.83**
goat	2.68	4.87	3.68	2.57	3.30	**2.00**
hiphop	**3.21**	8.08	8.02	4.62	6.94	3.37
horse	3.60	16.32	3.99	4.14	3.00	**2.62**
kongfu	**1.97**	2.51	5.42	3.71	5.78	3.28
park	**2.35**	5.89	3.95	3.49	3.33	2.93
pig	2.15	3.18	3.86	**2.08**	3.39	2.97
pot	**0.62**	1.25	0.94	1.49	0.80	0.70
skater	**4.72**	11.23	6.33	5.33	5.09	4.89
station	2.07	8.55	2.53	2.01	2.37	**1.53**
supertramp	9.68	9.76	14.70	8.99	17.40	**6.17**
toy	0.66	7.16	1.02	1.32	0.70	**0.58**
tricking	**4.23**	5.57	42.20	9.71	11.90	5.02
Average	**2.72**	6.77	6.19	3.72	4.33	2.78

Analysis

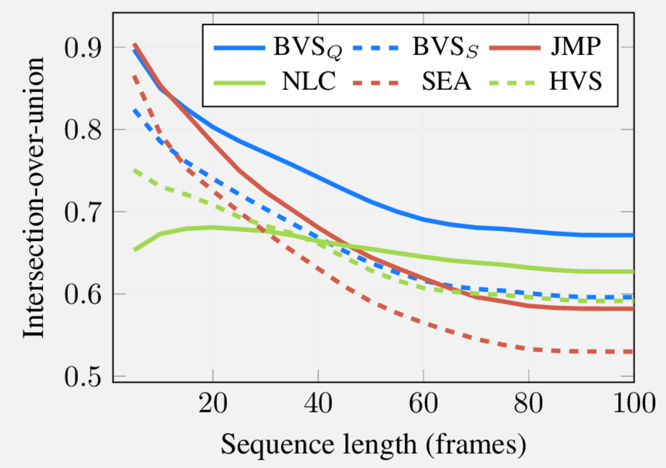
Parameters Evaluated two different sets of settings, one tuned for quality, **BVS_Q**, and the other for speed, **BVS_S**:

	BVS_Q (quality)	BVS_S (speed)
Feature space	YUV XY T	YUV XY T
Intensity grid size	35	15
Chroma grid size	30	10
Spatial grid size	$w/35, h/35$	$w/50, h/50$
Temporal grid size	2	2
Interpolation	Linear	Adjacent
Runtime	$0.37s$	$0.15s$

Running Time: per frame for a number of fast methods with code available:

	BVS_Q	BVS_S	SEA	JMP	NLC	HVS
480p	$0.37s$	$0.15s$	$6s$	$12s$	$20s$	$5s$
1080p	$1.5s$	$0.8s$	$30s$	$49s$	$20s$	$24s$

Temporal Decay:
IoU performance when a single mask is propagated. Our method degrades favorably when compared to other approaches.



Connectivity Analysis Mask propagation: on a pixel-level graph with increasing neighborhood sizes ω. Error decreases with larger neighborhoods but longer runtimes.