EpicFlow:
Edge-Preserving Interpolation of Correspondences for Optical Flow

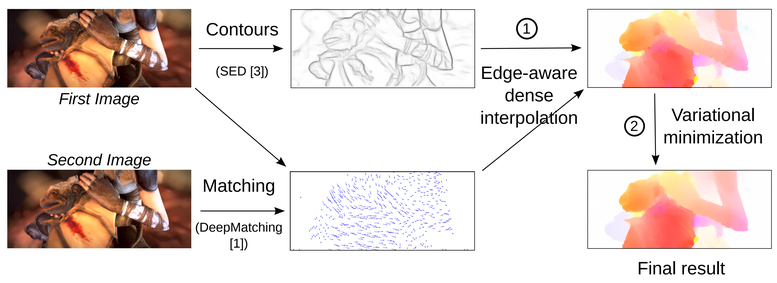
Jerome Revaud Philippe Weinzaepfel Zaid Harchaoui Cordelia Schmid

Inria Grenoble — informatics mathematics

Contributions

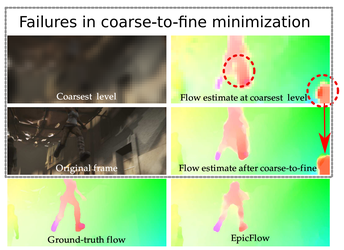
- A novel optical flow estimation method that:
  - leverages recent advances in **matching algorithms**
  - introduces an edge-aware **geodesic distance** that handles **motion discontinuities and occlusions**
  - yields state-of-the-art results
- Avoids coarse-to-fine minimization and works better for sufficiently dense matching
- Code available online at http://lear.inrialpes.fr/software

Overview of the method

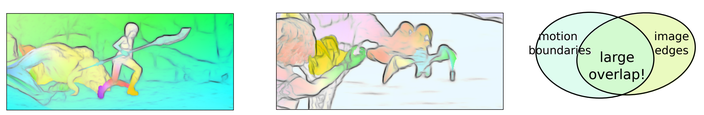


Motivations

- Variational approaches are generally minimized using a coarse to-fine scheme
- Drawbacks of coarse-to-fine:
  - edges and matches are not well-defined at coarse scales (thin parts might merge)
  - often unable to recover from errors that appeared at coarser scales
  - no theoretical guarantees or proof of convergence



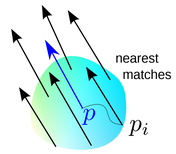
- **Main ideas**
  - Directly initialize the variational minimization with a dense interpolation of the matching computed at full image scale
  - We leverage the fact that motion boundaries and image edges coincide most of the time



  - Our interpolation is robust to motion discontinuities and occlusions due to using a geodesic distance on image edges

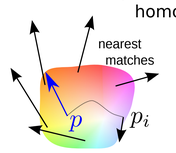
① Dense interpolation of an input matching

- Inputs to our method:
  1) a **matching** between the two images
     - obtained with DeepMatching [1] (≈ 5000 matches / image pair)
  2) the **contours** of the first image
     - obtained with the Structured Edge Detector (SED) [3]
- Interpolation procedure for every pixel *p*:
  1) Find the *K* nearest matches according to the geodesic distance (see below)
  2) Interpolation at pixel *p* using either:
     - **Nadaraya-Watson**:



simple **average** of nearest matches weighted by $e^{-\text{dist\_geodesic}(p,p_i)}$

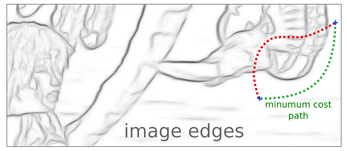
     - **Local affine fit**: (best for rotations & homographies)



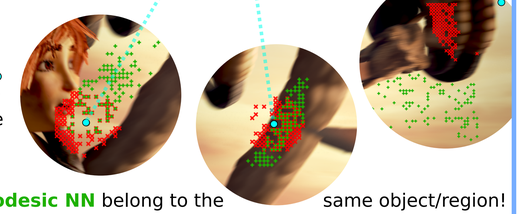
we solve a simple least-square problem (the match weight is the same than at left)

Edge-aware geodesic distance:

- Minimum cost among all possible paths between two points given a pixel-wise cost map
- Used to find the nearest matches of each pixel for local interpolation
- "**Edge-aware**" if the cost map is the output of an edge detector (e.g. SED [3])



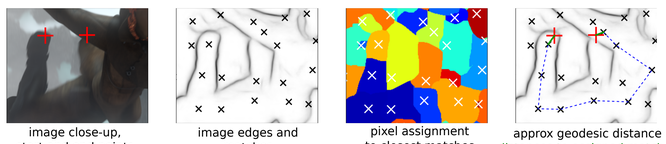
100 Nearest matches of ● retrieved with **Euclidean** (×) or **geodesic** (+) distance



Geodesic NN belong to the same object/region!

Fast geodesic distance approximation:

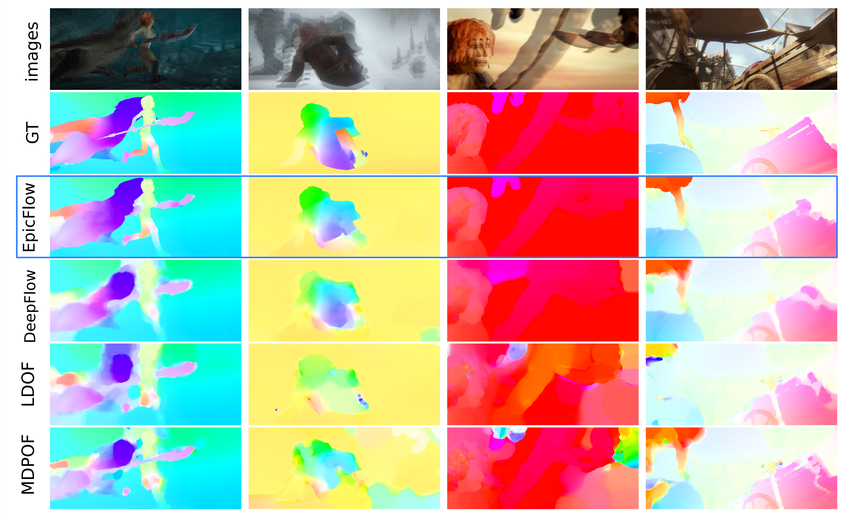
- Computed in a small graph instead of on all image pixels:
  - Graph nodes: matches in the first image
  - Each pixel is assigned to its closest match (using geodesic distance)
  - Djikstra's algorithm is used to compute geodesic distance in the graph
- No loss of performance compared to exact geodesic distances, 1000x speed-up



② Variational refinement

- **One-level** variational minimization:
  - initialization: dense matching interpolation
  - non-local smoothness term
  - classical solver (fixed point iterations, successive over relaxation)

Experimental results:



Impact of interpolators

- Local interpolation:
  - Nadaraya-Watson (**NW**) • Local affine (**LA**)
- Variational refinement: with/without

	Matching	Interpolator	MPI-Sintel	KITTI
inter.	DM	NW	4.143	5.460
	DM	LA	4.068	3.560
refine.	DM	NW	3.804	4.900
	DM	LA	**3.686**	**3.334**

Comparison with coarse-to-fine

for the same input matching and contours:

Flow method	MPI-Sintel	KITTI	Middlebury	Time
DM+coarse-to-fine	4.095	4.422	**0. 321**	25s
DM+EpicFlow	**3.686**	**3.334**	0.380	16.4s

Sensitivity to contours & matching

- Contours:
  - SED [3] • gPb • Canny • image gradient
- Matching:
  - DeepMatching (**DM**) [1]
  - Kd-tree assisted PatchMatch (**KPM**) [2]
- Geodesic distance:
  - Euclidean • geodesic (exact or approx.)

Matching	Contour	Distance	MPI-Sintel	Kitti	Time
KPM	SED	Geodesic (approx)	5.764	11.31	6.4s
DM	SED	Geodesic (approx.)	3.686	3.334	16.4s
DM	SED	Geodesic (exact)	**3.677**	**3.216**	20.4s
DM	-	Euclidean	4.617	3.663	40s
DM	gPb	Geodesic (approx.)	4.161	3.437	26s
DM	Canny	Geodesic (approx.)	4.551	3.308	16.4s
DM	$\|\nabla_x \mathcal{I}\|_2$	Geodesic (approx.)	4.061	3.399	16.4s
DM	GT boundaries	Geodesic (approx.)	3.588		

Comparison to the state of the art:

- MPI-Sintel:

Method	AEE	AEE-occ	s0-10	s10-40	s40+	Time
EpicFlow	**6.285**	**32.564**	**1.135**	**3.727**	**38.021**	**16.4s**
TF+OFM	6.727	33.929	1.512	3.765	39.761	~400s
DeepFlow	7.212	38.781	1.284	4.107	44.118	19s
S2D-Matching	7.872	40.093	1.172	4.695	48.782	~2000s
Classic+NLP	8.291	40.925	1.208	5.090	51.162	~800s
MDP-Flow2	8.445	43.430	1.420	5.449	50.567	709s
NLTGV-SC	8.746	42.242	1.587	4.780	53.860	
LDOF	9.116	42.344	1.485	4.839	57.296	30s

- KITTI:

Method	AEE-noc	AEE	Out-Noc 3	Out-All 3	Time
PH-Flow	**1.3**	2.9	5.76%	10.57	800s
BTF-ILLUM	1.5	**2.8**	6.52%	**11.03%**	80s
EpicFlow	1.5	3.8	7.88%	17.08%	16s
TGV2ADCSIFT	1.5	4.5	6.20%	15.15%	12s (GPU)
DeepFlow	1.5	5.8	7.22%	17.79%	17s
NLTGV-SC	1.6	3.8	**5.93%**	11.96%	16s (GPU)
Data-Flow	1.9	5.5	7.11%	14.57%	180s
TF+OFM	2.0	5.0	10.22%	18.46%	350s

Best published results!

References:

[1] **DeepFlow: Large displacement optical flow with deep matching.** P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. *In ICCV, 2013.*

[2] **Computing nearest-neighbor fields via propagation-assisted kd-trees.** K. He and J. Sun. *In CVPR, 2012.*

[3] **Structured forests for fast edge detection.** P. Dollar and C. L. Zitnick. *In ICCV, 2013*