RAFT - Recurrent All-Pairs Field Transforms

October 10th 2025

by Émulie Chhor

RAFT - Recurrent All-Pairs Field Transforms

arXiv:2003.12039v3 [cs.CV] 25 Aug 2020

RAFT: Recurrent All-Pairs Field Transforms for Optical Flow

Zachary Teed and Jia Deng

Princeton University
{zteed,jiadeng}@cs.princeton.edu

Abstract. We introduce Recurrent All-Pairs Field Transforms (RAFT), a new deep network architecture for optical flow. RAFT extracts per-pixel features, builds multi-scale 4D correlation volumes for all pairs of pixels, and iteratively updates a flow field through a recurrent unit that performs lookups on the correlation volumes. RAFT achieves state-of-the-art performance. On KITTI, RAFT achieves an F1-all error of 5.10%, a 16% error reduction from the best published result (6.10%). On Sintel (final pass), RAFT obtains an end-point-error of 2.855 pixels, a 30% error reduction from the best published result (4.098 pixels). In addition, RAFT has strong cross-dataset generalization as well as high efficiency in inference time, training speed, and parameter count. Code is available at `https://github.com/princeton-vl/RAFT`.

1 Introduction

Optical flow is the task of estimating per-pixel motion between video frames. It is a long-standing vision problem that remains unsolved. The best systems are limited by difficulties including fast-moving objects, occlusions, motion blur, and textureless surfaces.

Optical flow has traditionally been approached as a hand-crafted optimization problem over the space of dense displacement fields between a pair of images [21,51,13]. Generally, the optimization objective defines a trade-off between a *data* term which encourages the alignment of visually similar image regions and a *regularization* term which imposes priors on the plausibility of motion. Such an approach has achieved considerable success, but further progress has appeared challenging, due to the difficulties in hand-designing an optimization objective that is robust to a variety of corner cases.

Recently, deep learning has been shown as a promising alternative to traditional methods. Deep learning can side-step formulating an optimization problem and train a network to directly predict flow. Current deep learning methods [25,42,22,49,20] have achieved performance comparable to the best traditional methods while being significantly faster at inference time. A key question for further research is designing effective architectures that perform better, train more easily and generalize well to novel scenes.

We introduce Recurrent All-Pairs Field Transforms (RAFT), a new deep network architecture for optical flow. RAFT enjoys the following strengths:

- Published in 2020 by Zachary Teed and Jia Deng (Princeton University)
- Optical Flow Problem
- Not to be confused with the distributed consensus algorithm

Table of Contents

What is Optical Flow?

What is Optical Flow?

Goal: Estimate motion field between two consecutive frames ie get a velocity vector for every pixel

Where is Optical Flow used?

- Sports Analytics: Motion Analysis for performance improvement and injury prevention
- Computer Graphics: Stable Rendering and Scene Reconstruction
- Robotics and Autonomous Driving: Obstacle motion and collision detection

Problem Statement

Definition

Given two frames I1 and I2, predict a flow field $F(x, y) = (u(x, y), v(x, y))$, where (u, v) is the horizontal and vertical displacement of pixel (x,y).

Side Notes:

- Feature Matching vs Optical Flow
- Sparse vs Dense Optical Flow Methods

Problem Statement

Assumptions

- **Brightness Constancy**: Pixel intensity doesn't change from one frame to the next

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

- **Spatial Smoothness**: Neighboring pixel have the same motion

$$I(x + dx, y + dy, t + dt) \approx I(x, y, t) + I_x u + I_y v + I_t$$

Problem Statement

Optical Flow Constraint

$$I_x u + I_y v + I_t = 0$$

where:

$$\boxed{I_x = \frac{\partial I}{\partial x} \quad I_y = \frac{\partial I}{\partial y}}$$

Spatial Derivatives

$$\boxed{u = \frac{dx}{dt} \quad v = \frac{dy}{dt}}$$

Optical Flow

$$\boxed{I_t = \frac{\partial I}{\partial t}}$$

Temporal Derivative

The spatial and temporal derivatives I_x, I_y and I_t are known, so the goal of the classical models is to solve for u and v (aperture problem)

Taxonomy

Classical Methods (Early 1980s - Mid 2010s)

Horn-Schunk (1981)

Formulate optical flow as energy minimization problem solved with iterative relaxation

$$E(u, v) = \iint \underbrace{(I_x u + I_y v + I_t)^2}_{\text{Brightness Constancy}} + \underbrace{\alpha^2 \left(\|\nabla u\|^2 + \|\nabla v\|^2\right)}_{\text{Smoothness Term}} dx\, dy$$

which can be solved with the Euler-Lagrange equations:

$$I_x(I_x u + I_y v + I_t) + \alpha^2 \nabla^2 u = 0$$
$$I_y(I_x u + I_y v + I_t) + \alpha^2 \nabla^2 v = 0$$

Classical Methods (Early 1980s - Mid 2010s)

Horn-Schunk (cont.)

The previous PDEs can be solved with iterative relaxation until convergence

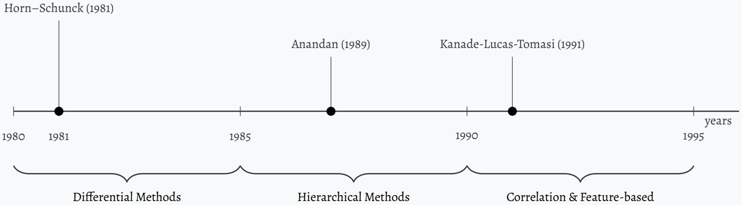
$$u^{k+1} = \bar{u}^k - \frac{I_x \left(I_x \bar{u}^k + I_y \bar{v}^k + I_t \right)}{\alpha^2 + I_x^2 + I_y^2}$$

$$v^{k+1} = \bar{v}^k - \frac{I_y \left(I_x \bar{u}^k + I_y \bar{v}^k + I_t \right)}{\alpha^2 + I_x^2 + I_y^2}$$

Classical Methods (Early 1980s - Mid 2010s)

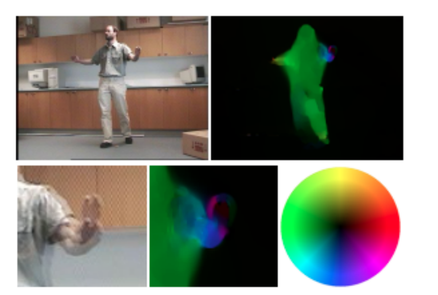
Classical Methods Improvements

1. Feature Extraction
2. Pixel Similarity
3. Iterative Relaxation

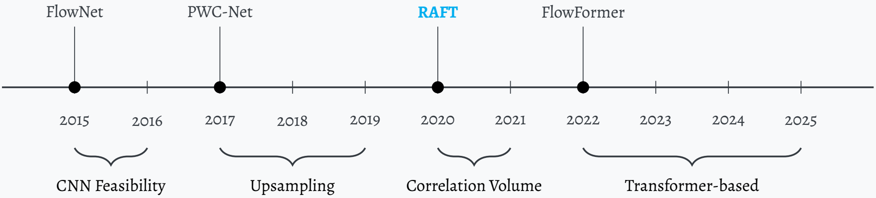


Classical Methods Challenges

- Accuracy: Large Displacement, Occlusions, Illumination & appearance changes
- Slow: no training phase, iteration has to be made at inference time



Deep Learning Methods (Mid 2010s-Present)



- FlowNet (162M): Showed that supervised learning could outperform classical methods, but heavy and struggled with large displacement
- PWC-Net (8.8M): Introduce pyramid, warping and cost volume ideas for upsampling (U-Net)
- RAFT (4.8M): All-pairs correlation volume
- FlowFormer (18.2M): Transformer-based model

Recurrent All-Pairs Field Transfoms (RAFT)

Architecture

Evaluation

Results

Input, Output and Flavors

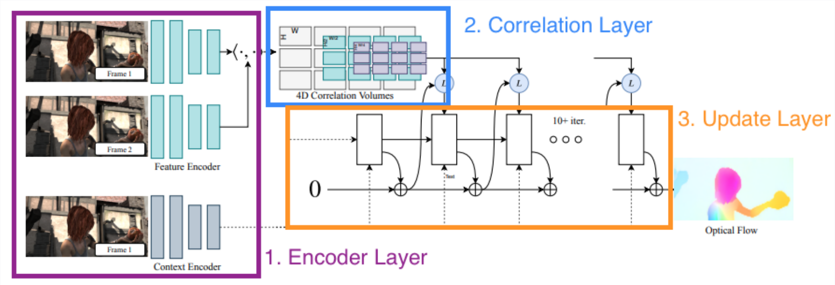
- Input: H x W x 2C; C=3
- Output: H x W x 2

Flavors:

- RAFT (4.8 M)
- RAFT-S (1M)

Architecture

Overview



Architecture

Novel Ideas

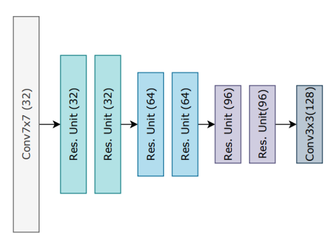
Limitations	**RAFT Improvements**
Fine details lost due to UNet architecture	Feature Encoder store and update a single flow field
Fixed number of iterations during training and inference	GRU-based opdate operator with shared weights
Large displacement missed because correlation performed locally	4D correlation volume

Architecture

Encoder Layer

The encoder layer is made of:

- Feature Encoder g_θ: extract per-pixel feature from I1 and I2
- Context Encoder h_θ: extract edges feature from I1



$$g_\theta : \mathbb{R}^{H\times W\times 3} \mapsto \mathbb{R}^{H/8\times W/8\times D}$$

Differences:

- Feature encoder uses instance normalization
- Context encoder uses batch normalization

Why RAFT add additional context encoder?

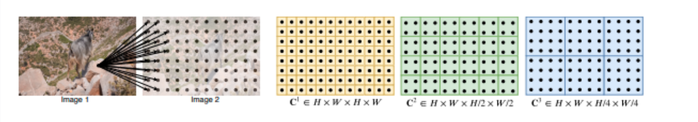
Architecture

Correlation Layer

Compute pixel similarity about small and large displacement:

1. **Correlation Pyramid**: For each pixel in I1, compute its similarity score to every pixel in I2 by taking the dot product between all pairs of feature vectors (or with a single matrix multiplication)
   This forms a 4-Layer Pyramid C^1, C^2, C^3, C^4, where C^k has dimension $HxWxH/2^k xW/2^k$ and k is the kernel size

$$C(i,j,k,l) = \sum_h g_\theta(I_1)_{ijh} \cdot g_\theta(I_2)_{klh}$$



Architecture

Correlation Layer (cont.)

2. **Correlation Lookup**: A lookup operator L_C which maps each pixel in I1 to its estimate correspondance in I2. Given the current predicted flow $f_1(u), f_2(v)$ and the pixel (x,y), compute the tentative correspondance $x^{'} = (x + f_1(x), y + f_2(y))$ and look in its local neighborhood to find the actual correspondance

$$N(x^{'})_r = \{x^{'} + dx \| \|dx\|_1 \leqslant r\}$$

Why it's more efficient:

1. Precompute correlation volume
2. Local lookup

Architecture

Update Layer

Instead of one-shot flow estimate like earlier model, RAFT uses a recurrent update mechanism:

1. **Initialization**: Zero optical flow ie assume no motion
2. **Inputs**: concatenate correlation features + flow features + context features into a single feature map

Update Layer (cont.)

3. **Update**: ConvGRU Refine flow step by step by integration correlation and context iteratively

$$z_t = \sigma(Conv_{3\times 3}([h_{t-1}, x_t], W_z)) \quad \text{(Update gate)}$$

$$r_t = \sigma(Conv_{3\times 3}([h_{t-1}, x_t], W_r)) \quad \text{(Reset gate)}$$

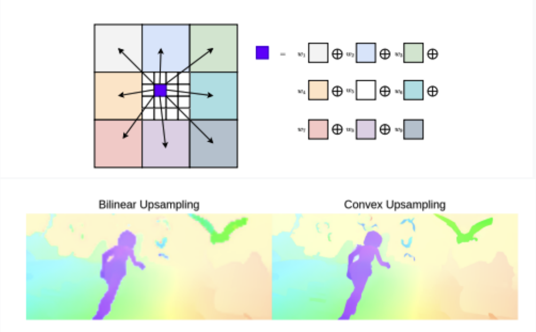
$$\tilde{h}_t = \tanh(Conv_{3\times 3}([r_t \odot h_{t-1}, x_t], W_h)) \quad \text{(Candidate hidden state)}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad \text{(New hidden state)}$$

4. **Flow Prediction**: Flow prediction is updated $f^{k+1} = f_k + \Delta f$ and prediction are made at 1/8 resolution for image efficiency

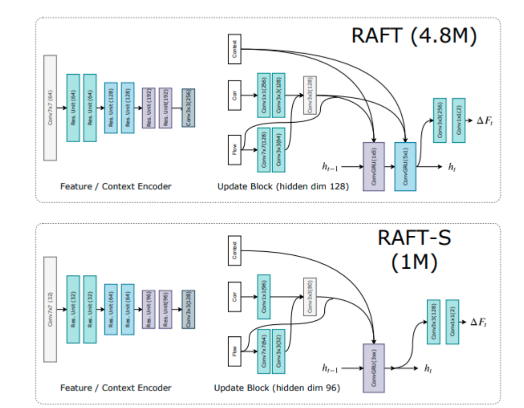
Update Layer (cont.)

5. **Upsampling**: Use a learn convex mask over 3x3 neighborhood instead of bilinear upsampling



RAFT Flavors

Differences



Differences:

- RAFT-S uses bottleneck residual units and 3x3 convolution in GRU
- Full model uses GRU updates blocks with 1x5 and 5x1 filters to increase receptive field

Evaluation

Datasets

- Hardware: 2x 2080Ti GPU
- Datasets:
  - FlyingThings: Synthetic dataset with large displacement and complex occlusions
  - KITTI: Real driving scene images with sparse ground truth from LiDAR. Involves rigid motion
  - Sintel: Synthetic animated movie with non-rigid motion

Sintel Example

FlyingThings Example

Evaluation

Training

1. Pretrained on FlyingThings for 100K iterations
2. Trained on FlyingThings3D for 100K iterations
3. Fine-tuned on Sintel and KITTI for 100K iterations

Intuition:

- Use FlyingThings to learn priors because is diverse
- Evaluate using both synthetic dataset and real dataset

Evaluation

Metrics

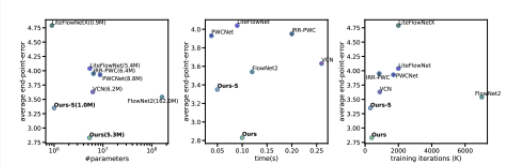
- Loss Function: L1 distance with exponentially increasing weight

$$\mathcal{L} = \sum_{i}^{N} \gamma^{N-i} \|f_{gt} - f_i\|_1; \gamma = 0.8$$

- Optimizer: AdamW with gradient clip $[-1, 1]$
- Metric: End-Point-Error (EPE)

Results

- Training Iterations
  - Surpass PWC-Net after 6 updates
  - Surpass FlowNet after 3 updates
- EPE on Sintel: 5.04 vs 8.36 (40% reduction)
- EPE on KITTI: 2.855 vs 4.098 (16% reduction)
- Inference Time: Doesn't beat PWC-Net (0.034s)
  - RAFT-S: (0.043s)
  - RAFT: (0.097s)



Perspective

Limitations

- Non-rigid motion and occlusions not handled well
- Still requires O(MN) to compute correlation volume
- Large displacement bottlenecked by higher pyramid-level
- Model is still large

Future Directions

- SEA-RAFT: reduce runtime, faster convergence, better generalization
- FlowFormer: uses transformers to improve upon non-repetitive larger displacement and non-rigid motion
- \ours: occlusion mask with forward-backward consistency check
- GMFlow: reduce dependence on many iterations

Questions?

Sources

1. Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in Proc. ECCV, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Lecture Notes in Computer Science, vol. 12347, pp. 402–419, 2020
2. E. Meinhardt-Llopis, J. Sánchez, and D. Kondermann, "Horn–Schunck optical flow with a multi-scale strategy," Image Processing On Line, vol. 3, pp. 151–172, Jul. 2013, doi:10.5201/ipol.2013.20
3. T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," in Proc. IEEE CVPR, 2009.

Sources (cont.)

4. D. Sun, C. Herrmann, F. Reda, M. Rubinstein, D. Fleet, and W. T. Freeman, "What Makes RAFT Better Than PWC-Net?", 2022.
5. J. Jeong, H. Cai, R. Garrepalli, J. M. Lin, M. Hayat, and F. Porikli, "OCAI: Improving Optical Flow Estimation by Occlusion and Consistency Aware Interpolation," arXiv preprint arXiv:2403.18092v1, 2024.
6. M. Zhai, X. Zuezhi, "Geometry Understanding from Autonomous Driving Scenarios Based on Feature Refinement," 2021.
7. Y. Yu, M. Liu, "Split-Attention Multiframe Alignment Network for Image Restoration", 2020.