Learning Correspondences

CS280

April 7 2025

Angjoo Kanazawa

Logistics

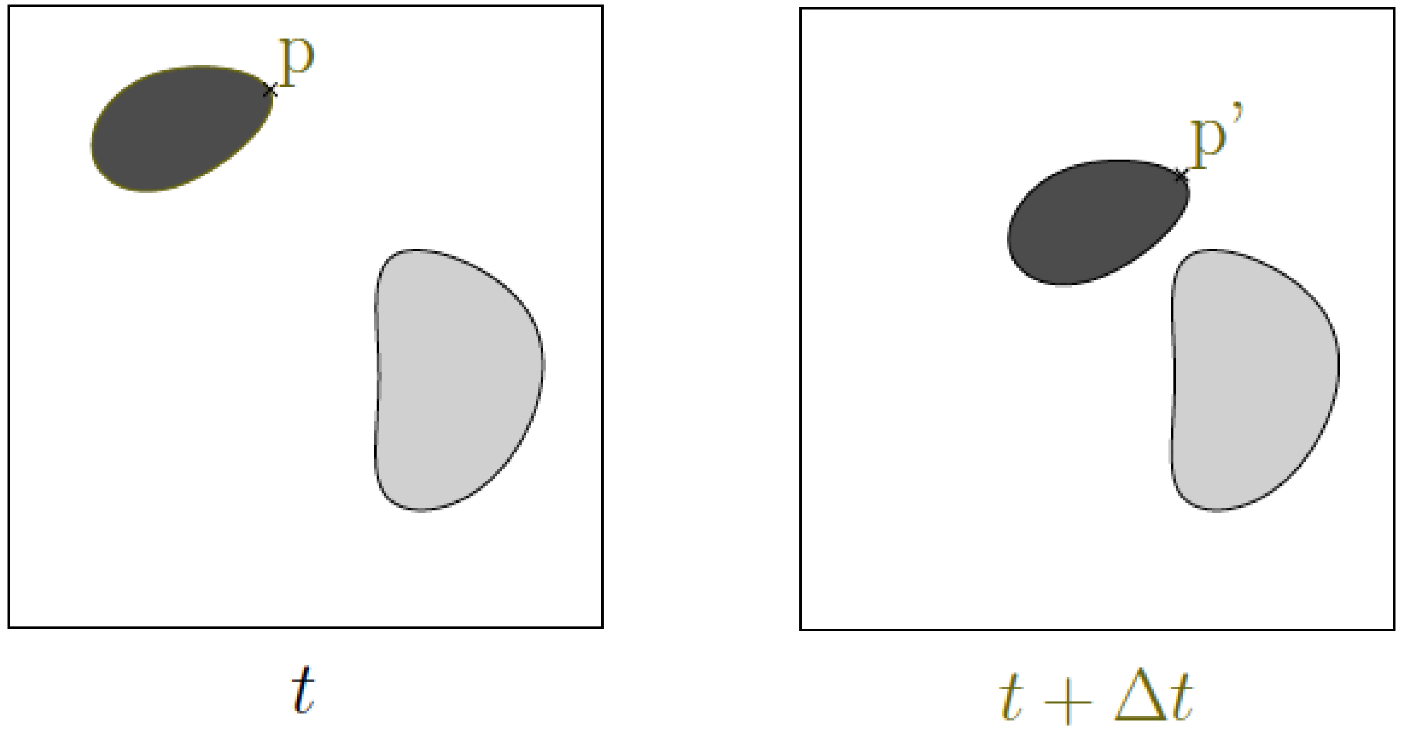
- Project Proposal write up due this Friday
- Midterm grading is almost over
- Hw3 due 4/14

Takeo Kanade

What are the three most important problems in computer vision?

“Correspondence, Correspondence, Correspondence!”

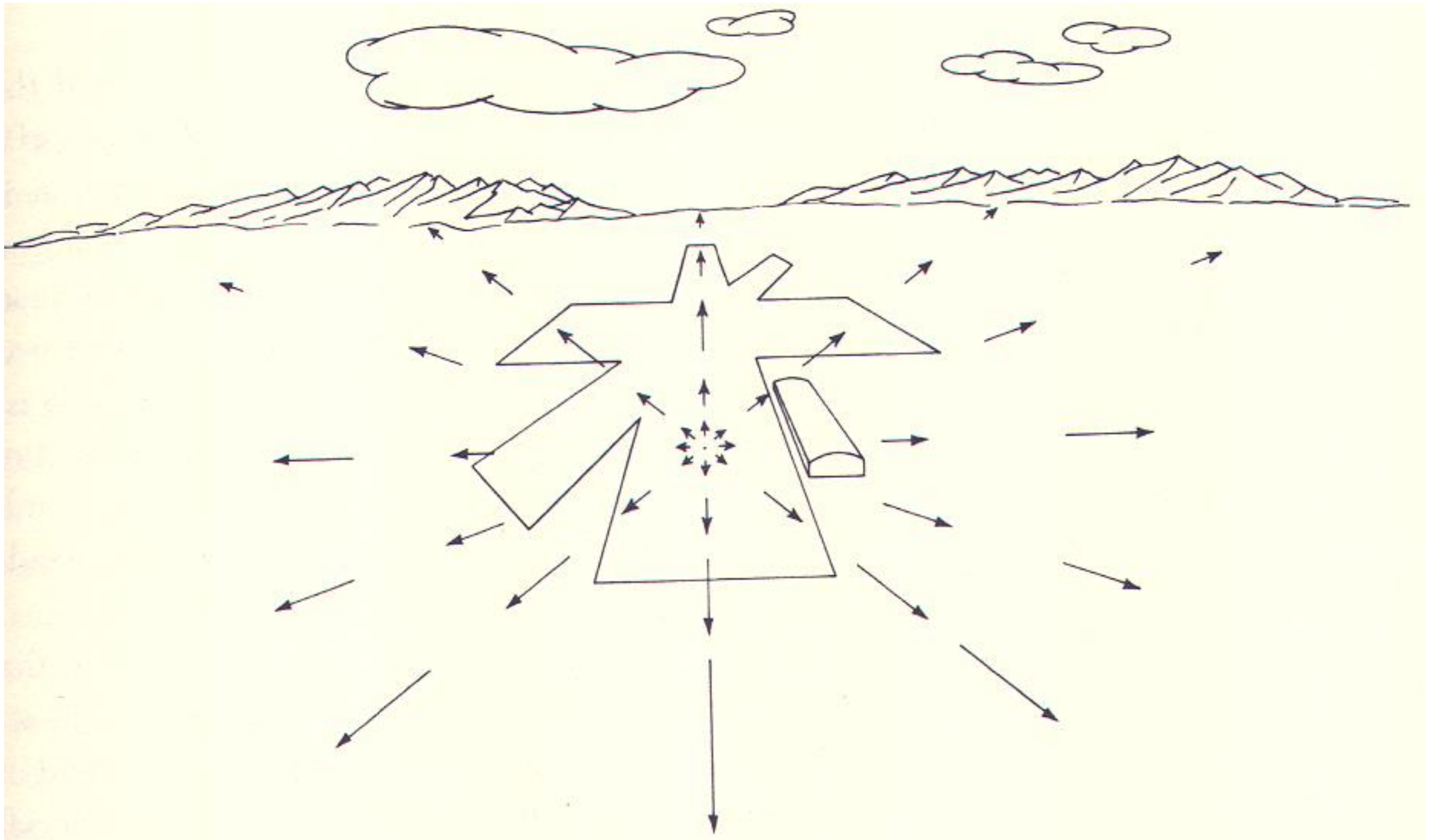
Optical flow is based on correspondence over time



If we can solve this correspondence, then if p has coordinates (x_1, y_1, t_1) and p' has coordinates (x_2, y_2, t_2), we get $u = \frac{x_2 - x_1}{t_2 - t_1}$ and $v = \frac{y_2 - y_1}{t_2 - t_1}$

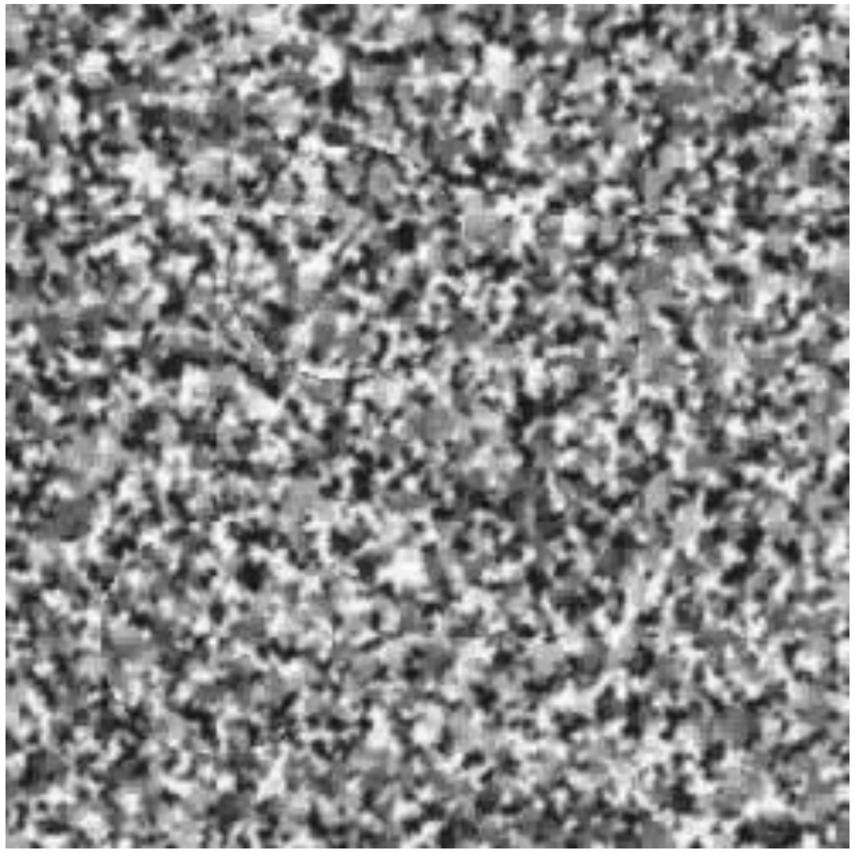
Slide credit: Jitendra Malik

Gibson’s example I:
Optical flow for a pilot landing a plane



Motion is a powerful perceptual cue

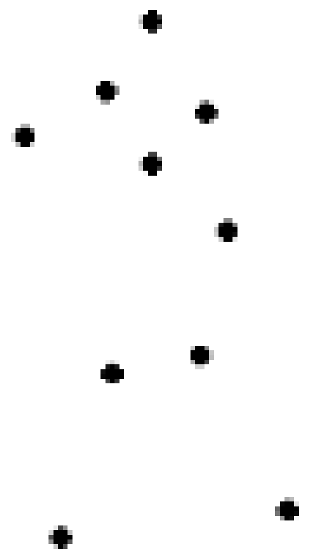
- Sometimes, it is the only cue



Slide credit: Lana Lezebnik

Motion is a powerful perceptual cue

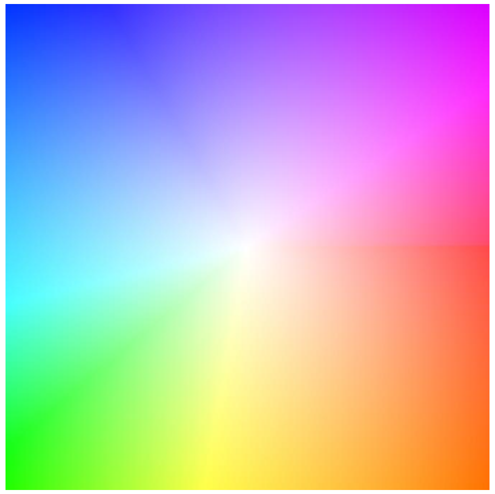
- Even “impoverished” motion data can evoke a strong percept



G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis",
Perception and Psychophysics 14, 201-211, 1973.

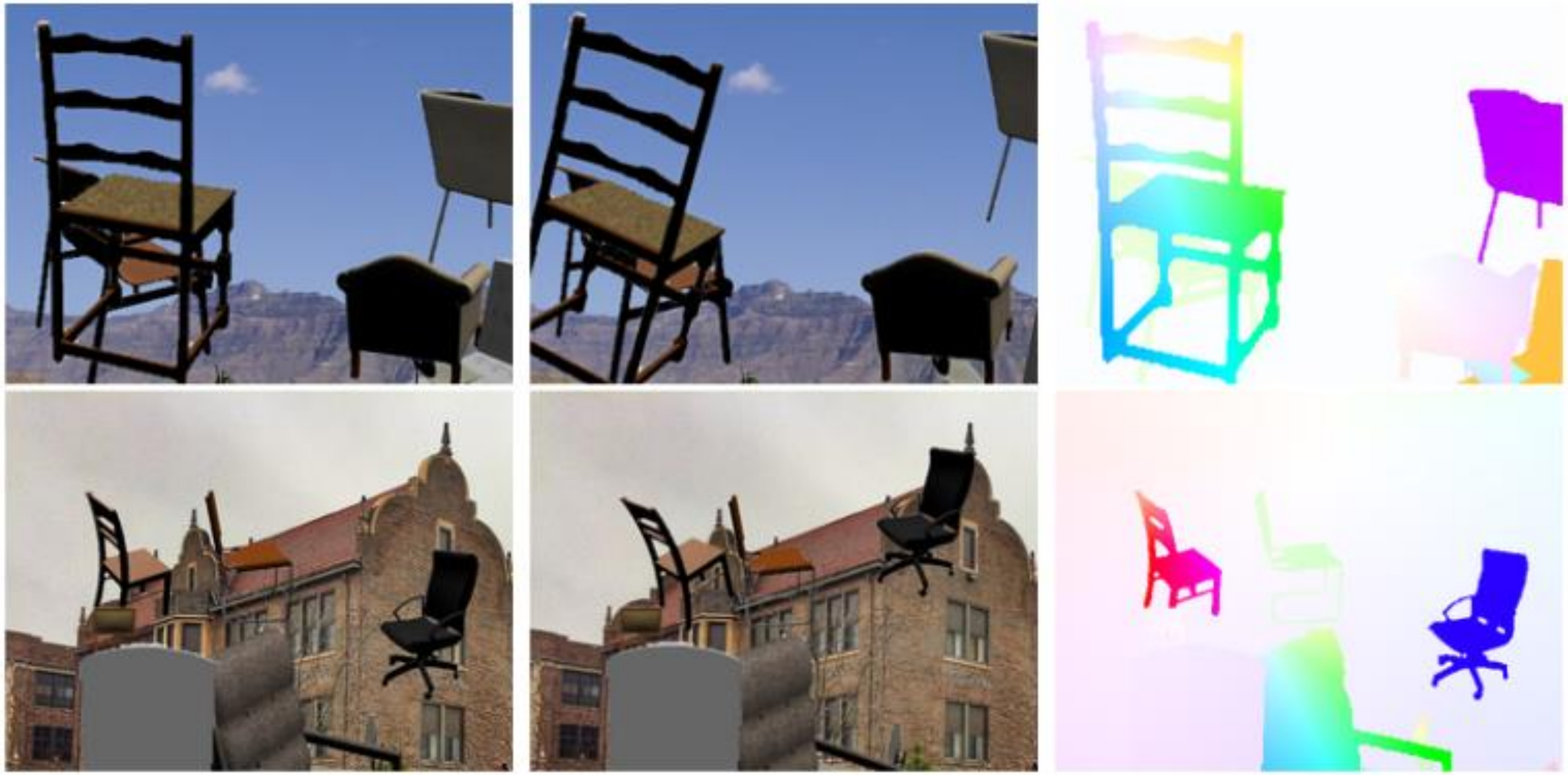
Slide credit: Lana Lezebnik

Sintel


Color Code

Butler et al. ECCV 2012

Flying Chairs



Dosovitskiy, ICCV 2015 FlowNet

Predicting Flow

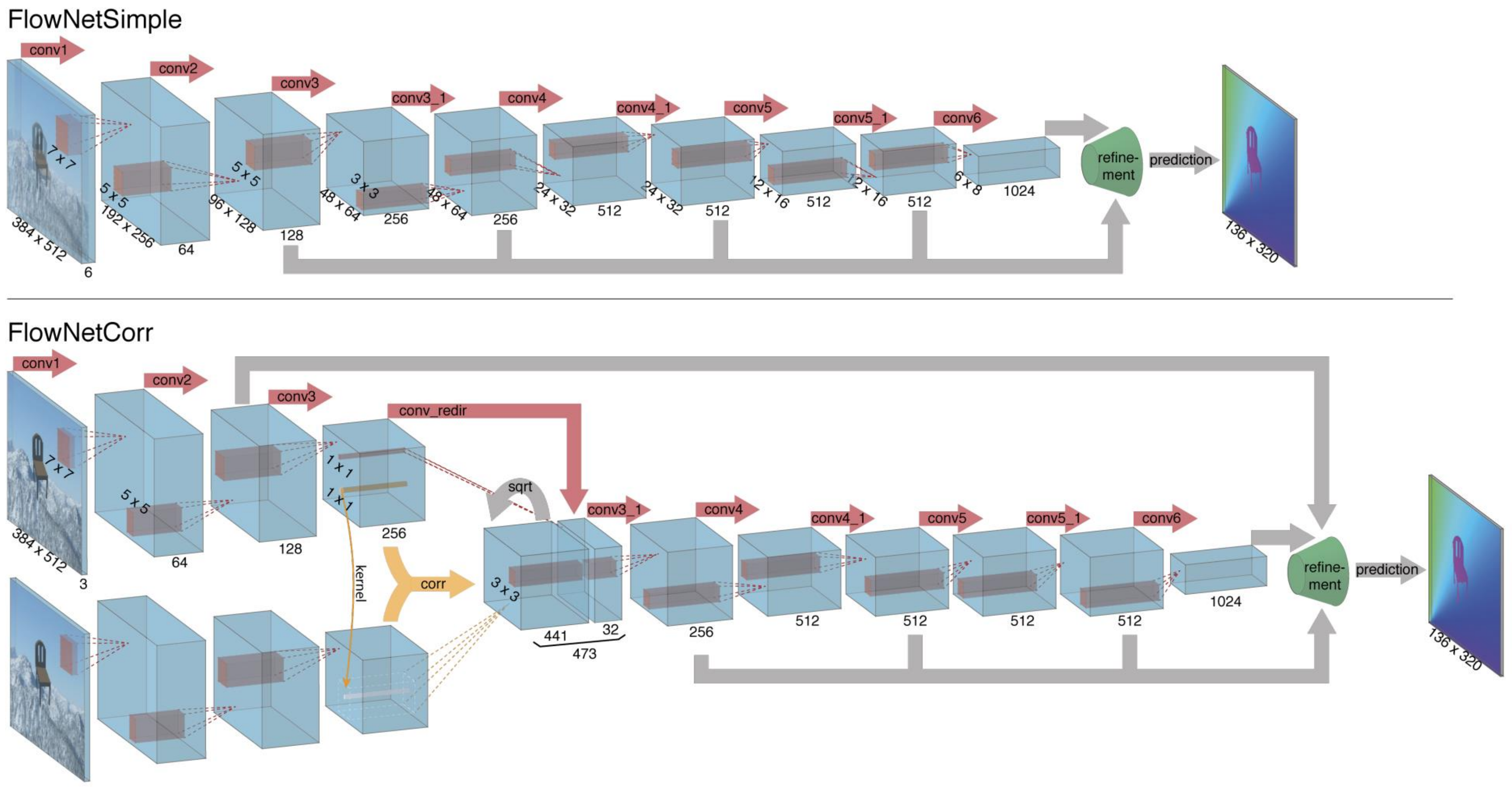
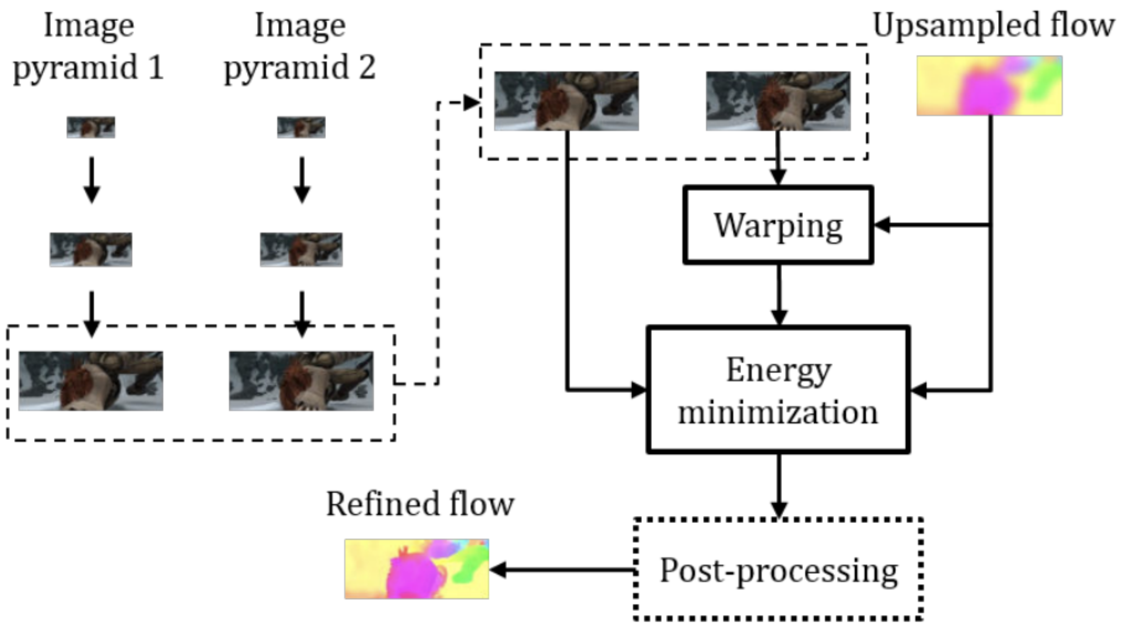


Figure 2. The two network architectures: FlowNetSimple (top) and FlowNetCorr (bottom).

Dosovitskiy, ICCV 2015 FlowNet

Sun et al. CVPR 20180

Classic VS Learned (PWC-Net)



Traditional Coarse-to-Fine

Big Picture: RAFT Series (RAFT, RAFTStereo, DROID-SLAM, RAFT3D)

Works really well! We’ve been using them a lot

What does it do? 2 Core ideas in RAFT and DROID-SLAM:

1. **(RAFT) learning to update, based on features conditioned on current estimate**
2. **(DROID-SLAM) Solve an optimization problem, but the objective fn contains target that is an output of a NN, which is conditioned by the current estimate**

- They are all supervised with synthetic data, synthetic to real generalization works bc of what get’s sent to NN
- All with a very efficient implementation & insights, very impressive

RAFT (Recurrent All-pairs Field Transformation)

- Teed and Deng et al. ECCV 2020 Best paper Award

- Works really well!

- On Sintel “RAFT obtains an end-point-error of 2.855 pixels, a 30% error reduction from the best published result (4.098 pixels).”

- Input: H x W img1, img2
- Output: dense flow H x W x 2

Step 1: compute features

Input: img1, img2

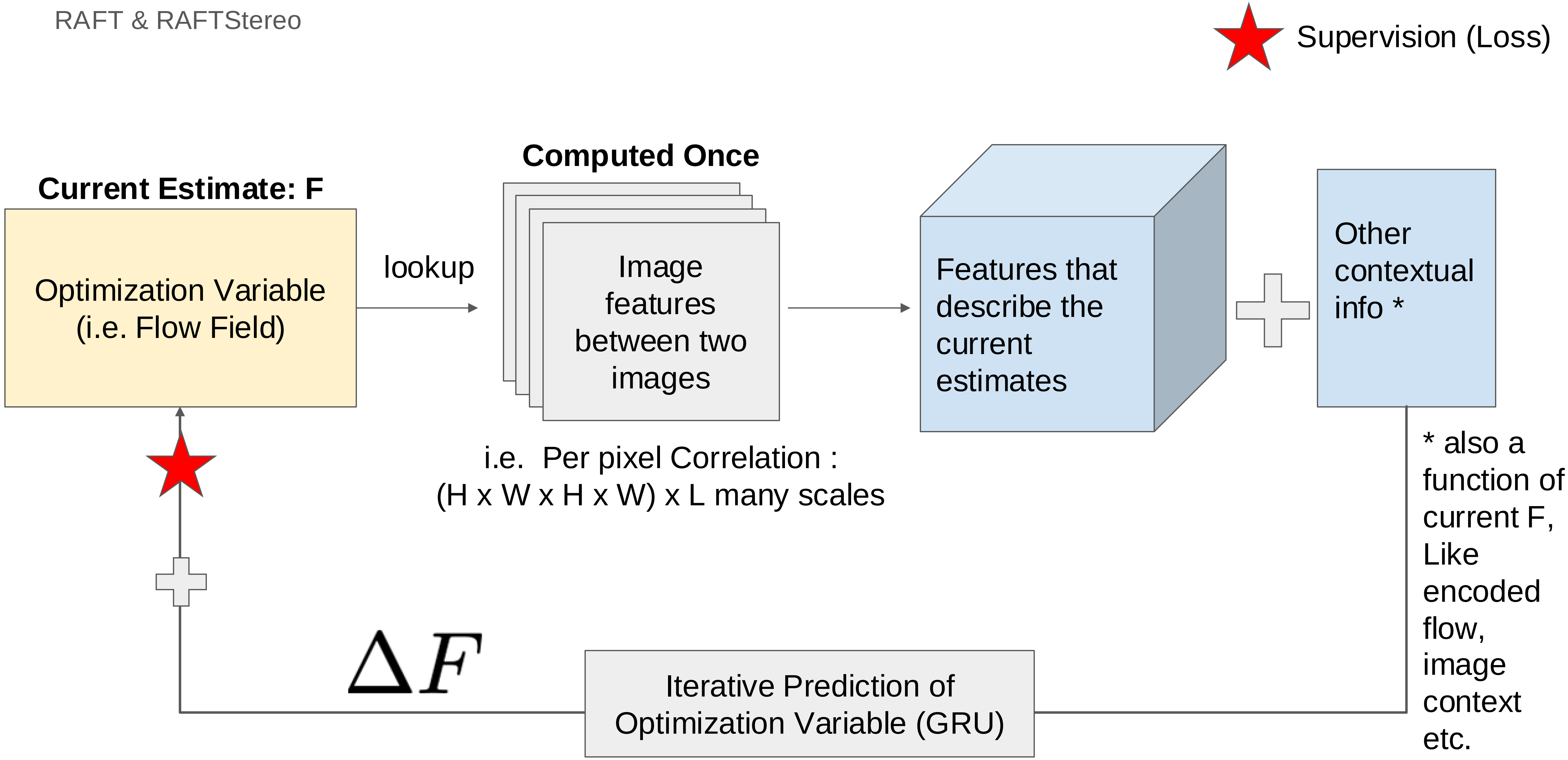
Compute image features from both once!

→ H/8 x W/8 x D

Compute a correlation volume H x W x H x W once

Concept 1: Feature Look up based on current (flow) estimates

RAFT & RAFTStereo

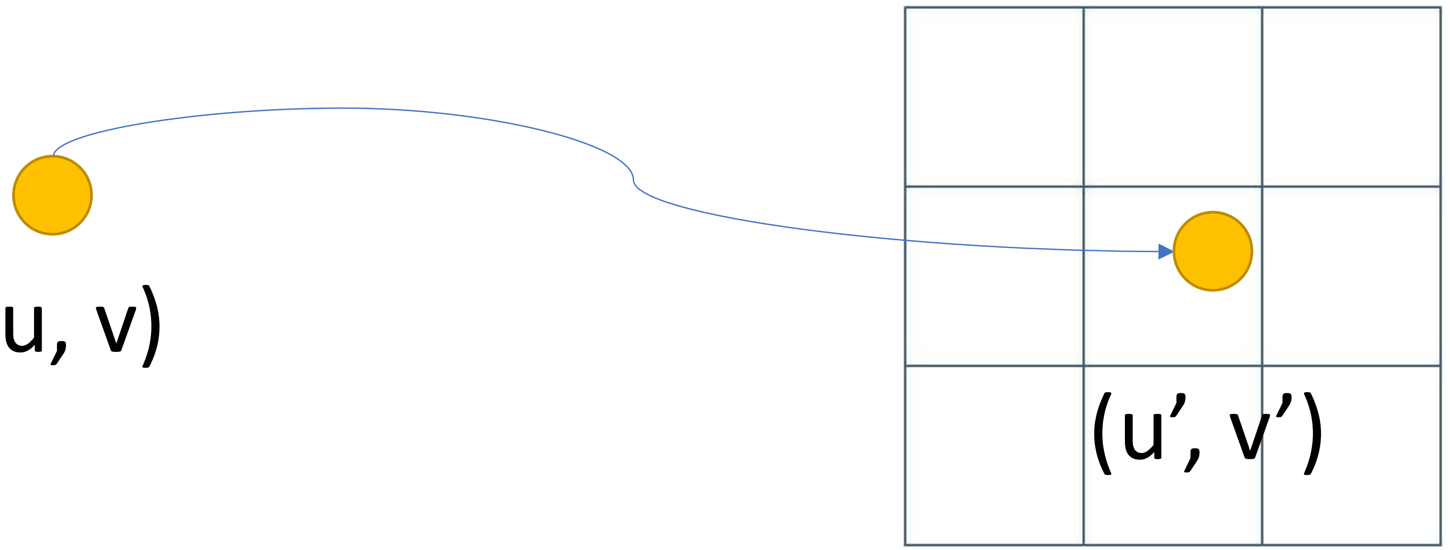


Supervision (Loss)
Current Estimate: F
Optimization Variable (i.e. Flow Field)
lookup
Computed Once
Image features between two images
i.e. Per pixel Correlation : (H x W x H x W) x L many scales
Features that describe the current estimates
Other contextual info *
* also a function of current F, Like encoded flow, image context etc.
ΔF
Iterative Prediction of Optimization Variable (GRU)

Look up into the correlation volume

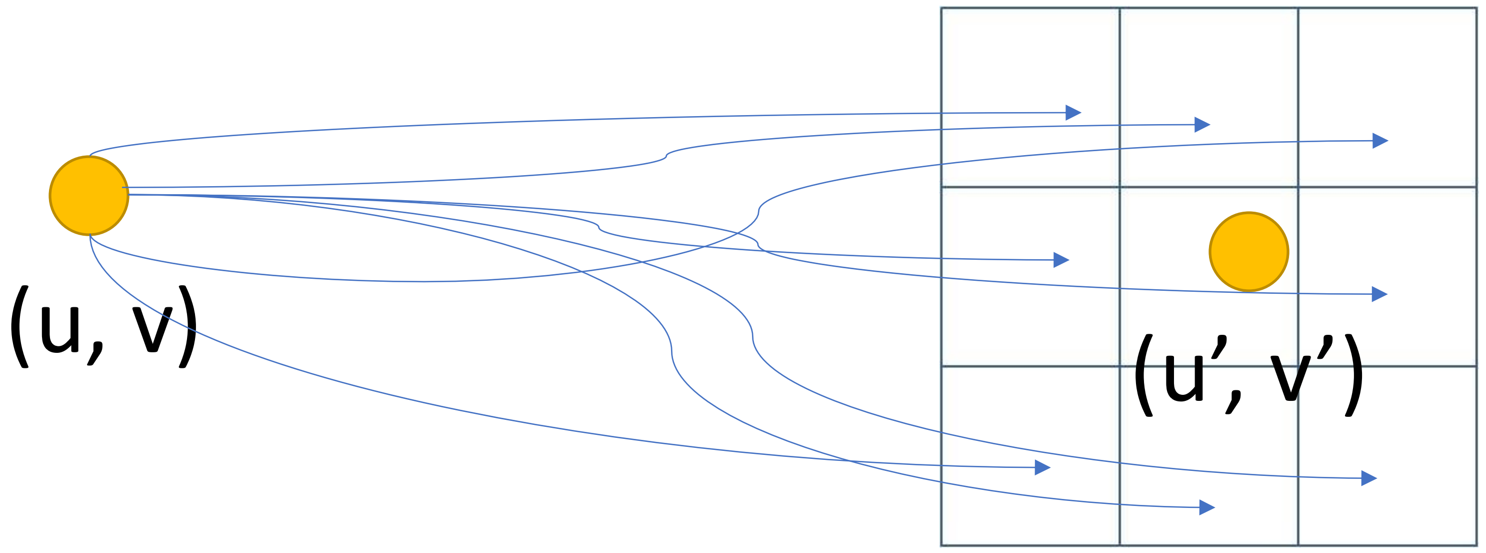
- For pixel (u, v) in img1, (u’, v’) = (u + f_u, v + f_v) is the current correspondence in img2 via flow (f_u, f_v)

- look up correlation between these: correlation[u, v, u’, v’]



Look up into the correlation volume

- Also look up around the neighbor grids with radius r : (2r+1) x (2r+1)



After look up

You have H x W x 2 flow

Use that to look up the volume → results in H x W x |grid|

Do this over multiple scales: H x W x |grid| * L

Correlation Volume:

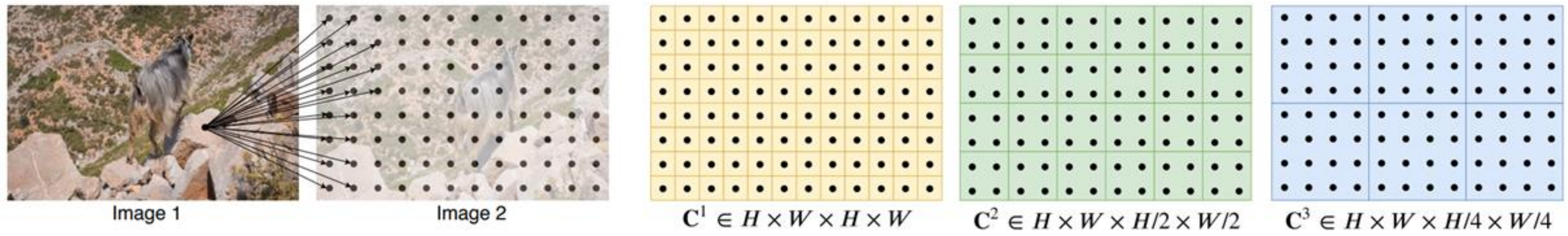
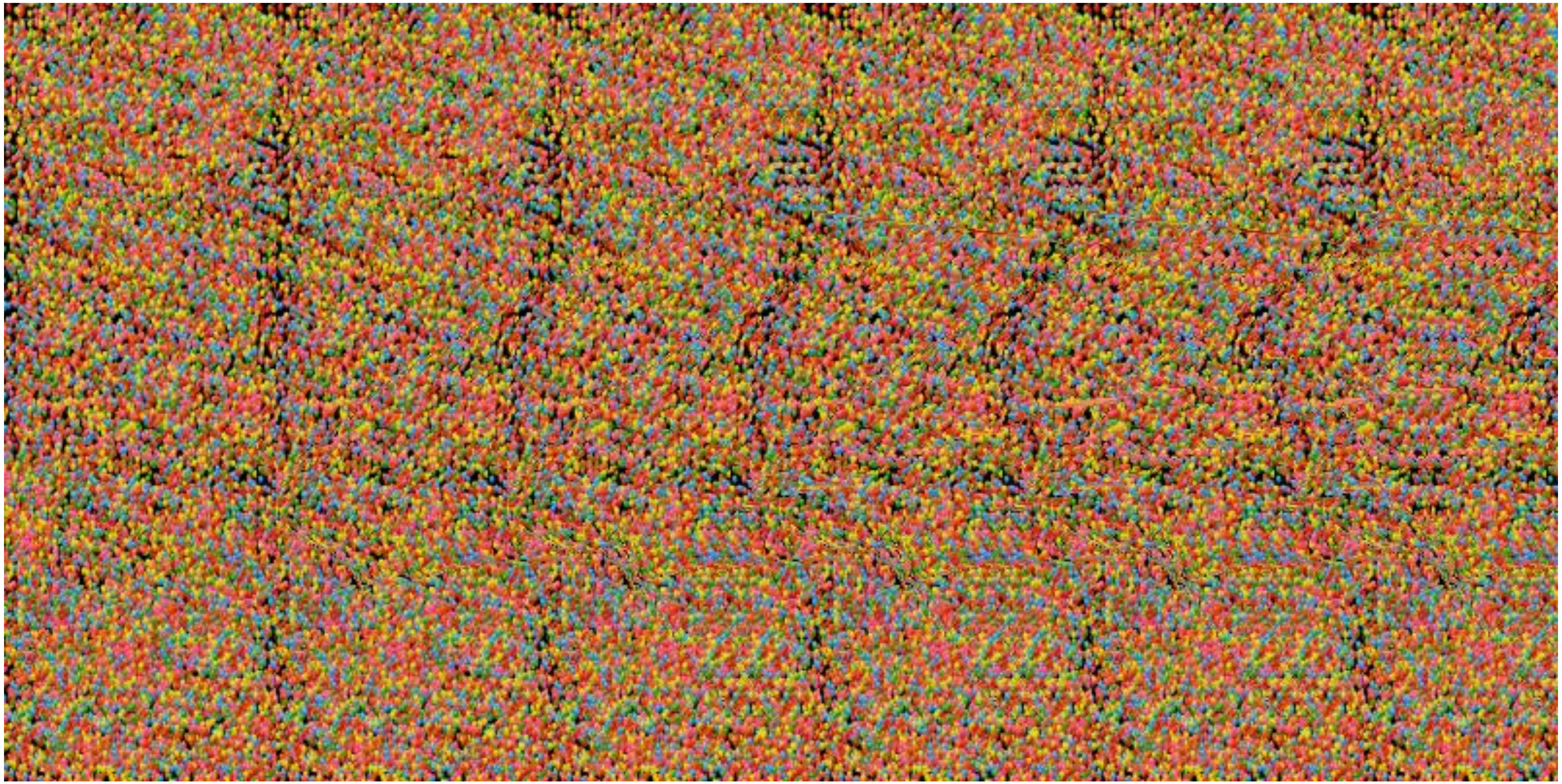


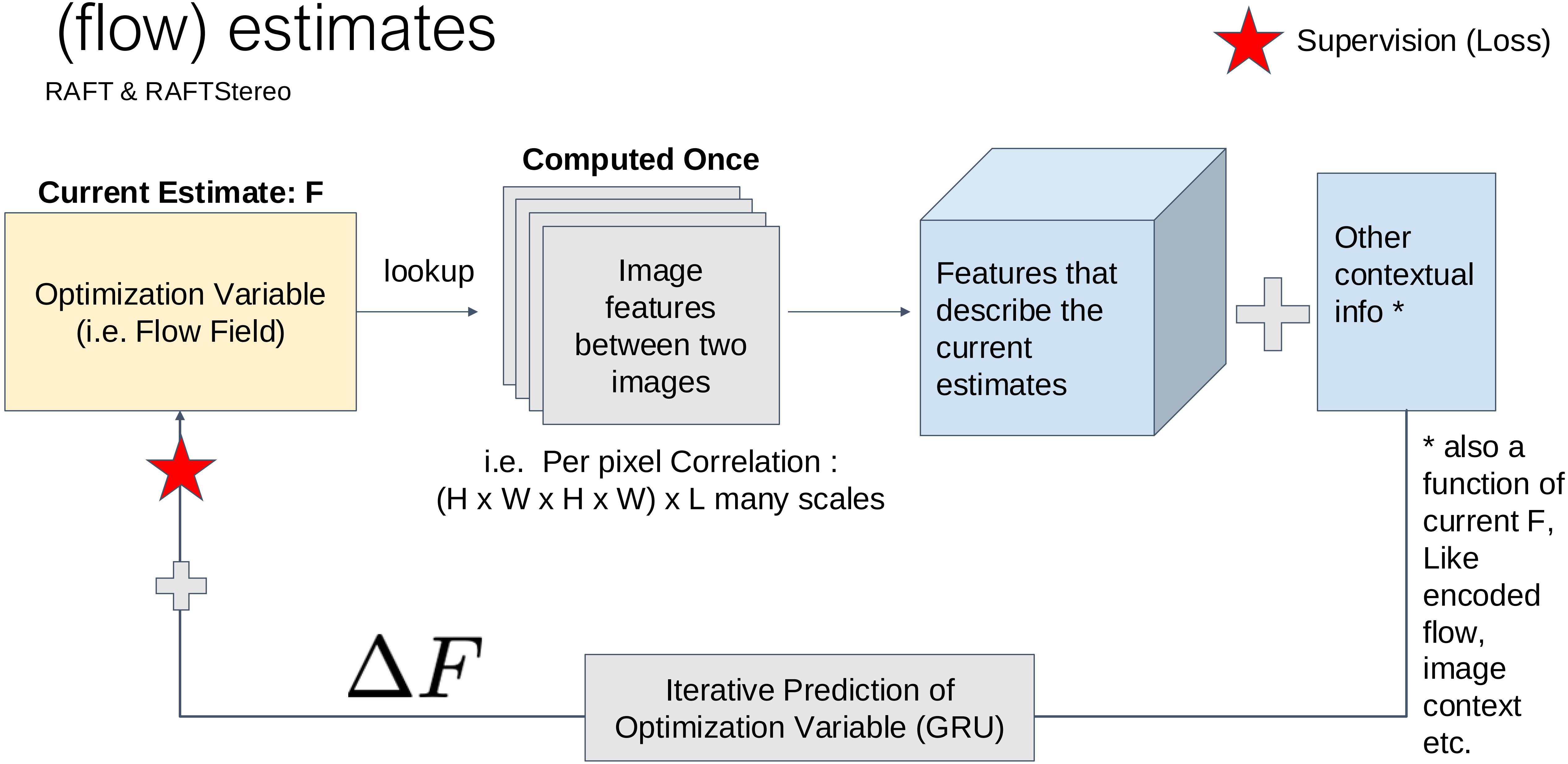
Image 1
Image 2
$\mathbf{C}^1 \in H \times W \times H \times W$
$\mathbf{C}^2 \in H \times W \times H/2 \times W/2$
$\mathbf{C}^3 \in H \times W \times H/4 \times W/4$

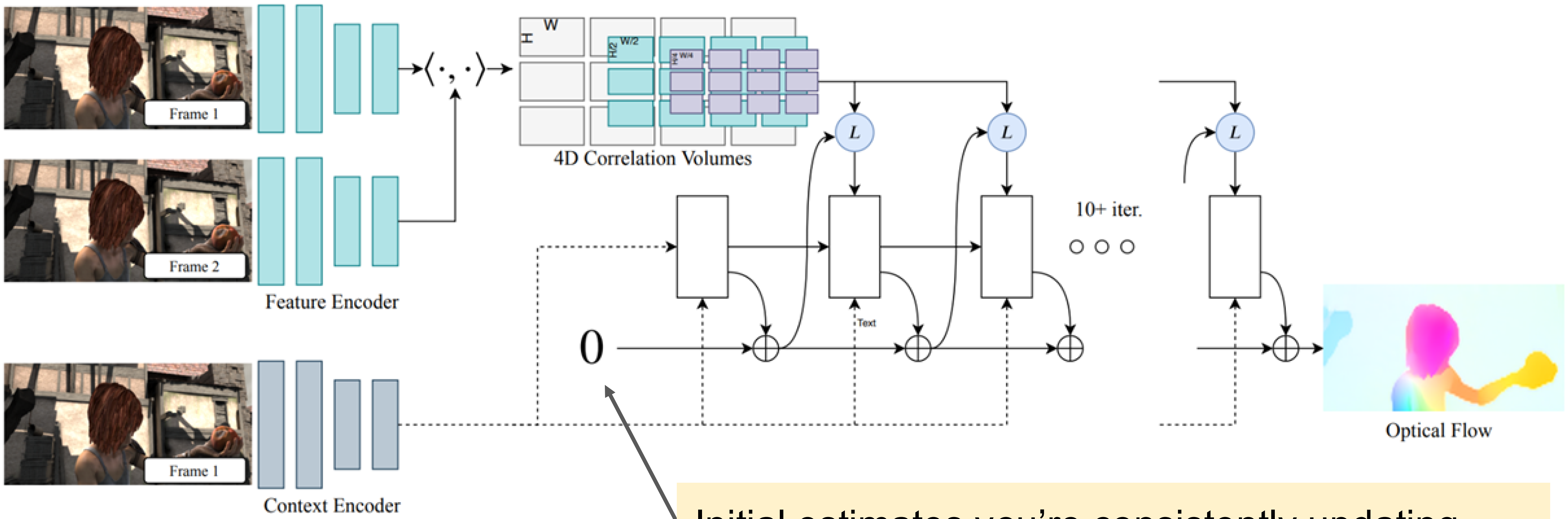
Correlation feature intuition



Concept 1: Feature Look up based on current (flow) estimates

RAFT & RAFTStereo


Supervision (Loss)
Computed Once
Current Estimate: F
Optimization Variable (i.e. Flow Field)
lookup
Image features between two images
Features that describe the current estimates
Other contextual info *
i.e. Per pixel Correlation : (H x W x H x W) x L many scales
* also a function of current F, Like encoded flow, image context etc.
ΔF
Iterative Prediction of Optimization Variable (GRU)



Initial estimates you're consistently updating

Question

This is supervised learning, trained on synthetic data!

Why does generalize so well (does not suffer from as much domain gap)?

Reasons

1. not dependent on the statistics of the underlying RGB texture bc it’s looking at the correlation *

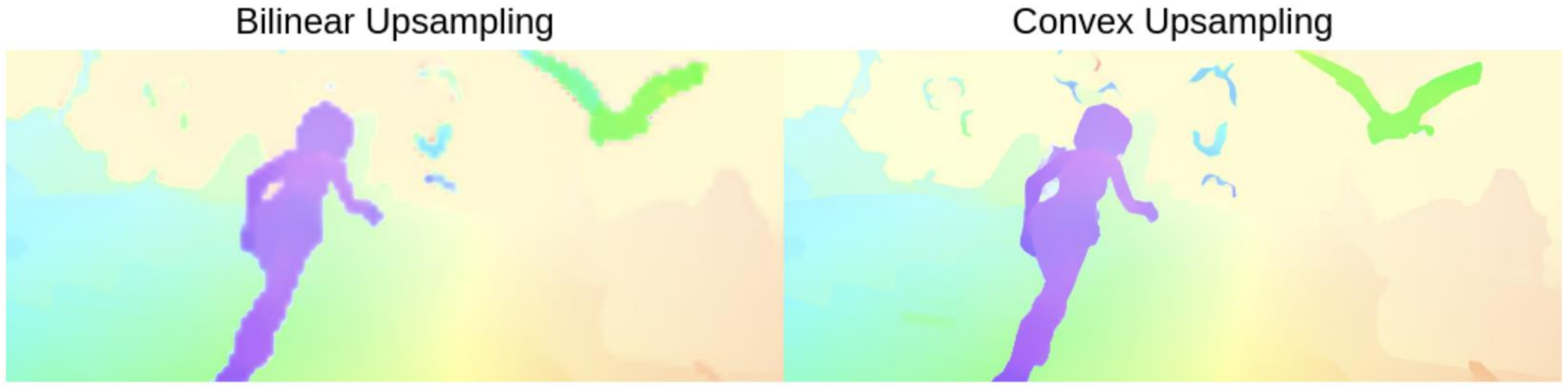
2. It output the residual: how to move to make it better, a single update → iterative, not predicting the final flow directly

Learning to update (”one step of an optimization algorithm”) given “features” that corresponds to the current estimates

Extra nice ideas

- Fast computation of the correlation volume
  - instead of computing O(HWHW) once, you can reduce this to O(HWM), where M is the number of runtime iterations (~32<<HW)

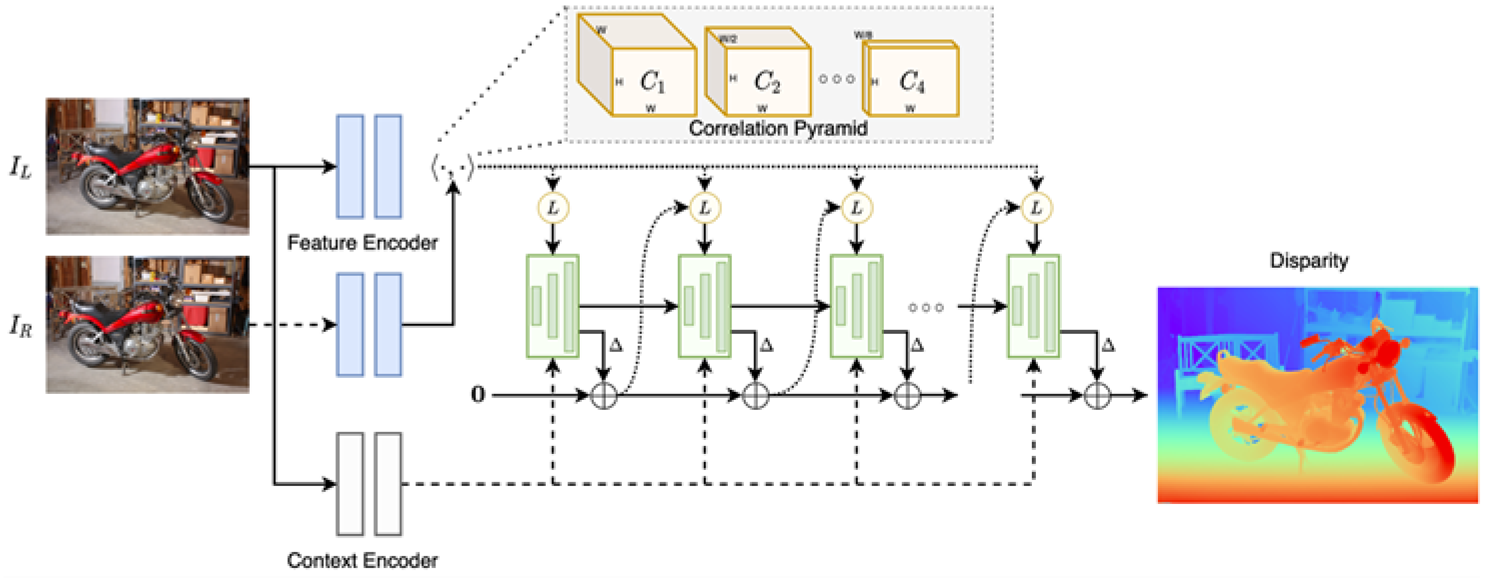
- Predicted upsampling weights (H/8xW/8 -> H x W via H/8×W/8×(8×8×9) weights)



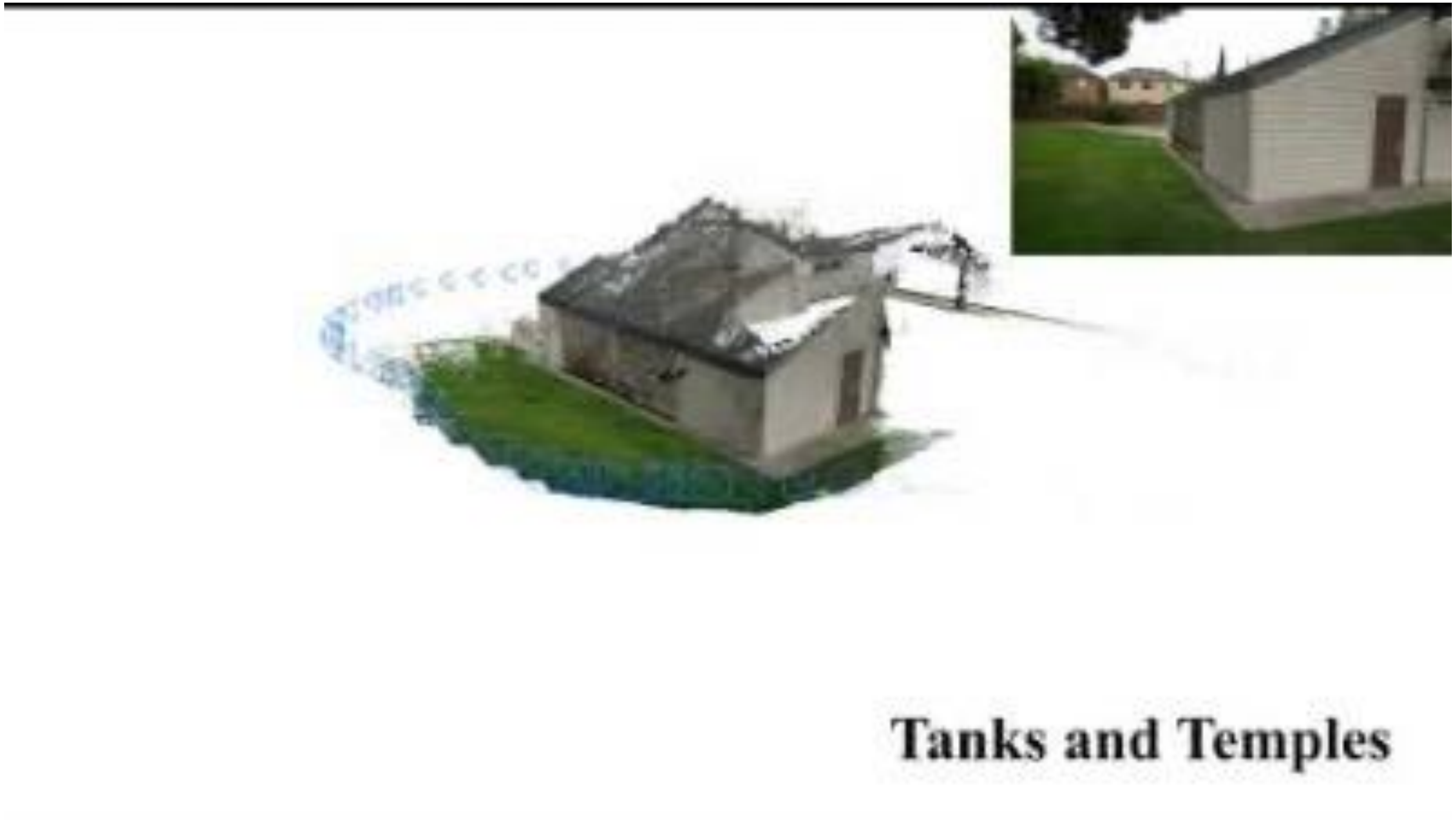
RAFTStereo: Correspondence is limited to horiz line

So correlation vol is now: H x W x W

Rest is pretty much the same (easier problem then Flow) mod implementation details



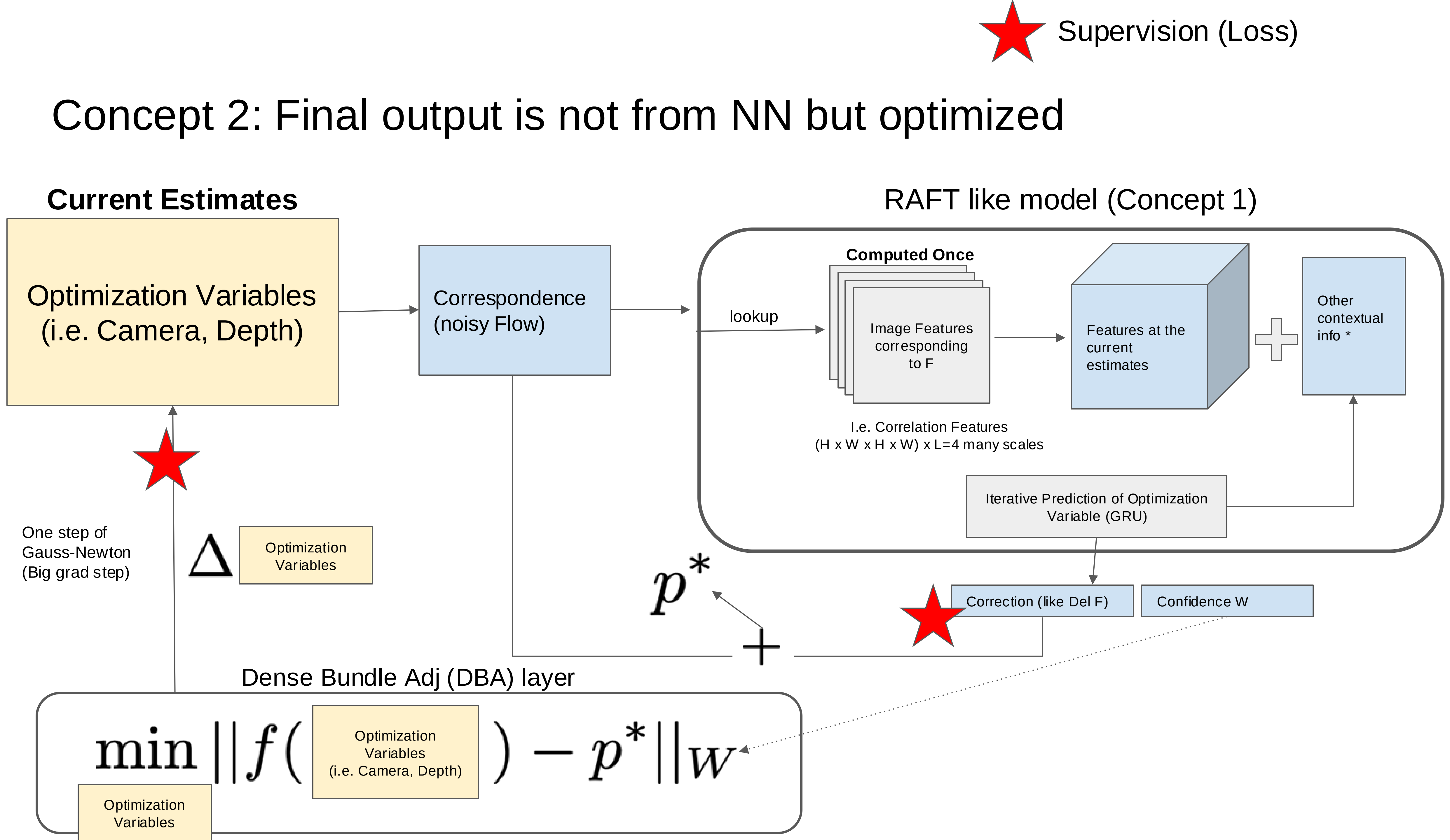
DROID-SLAM Teed and Deng, NeurIPS 2021



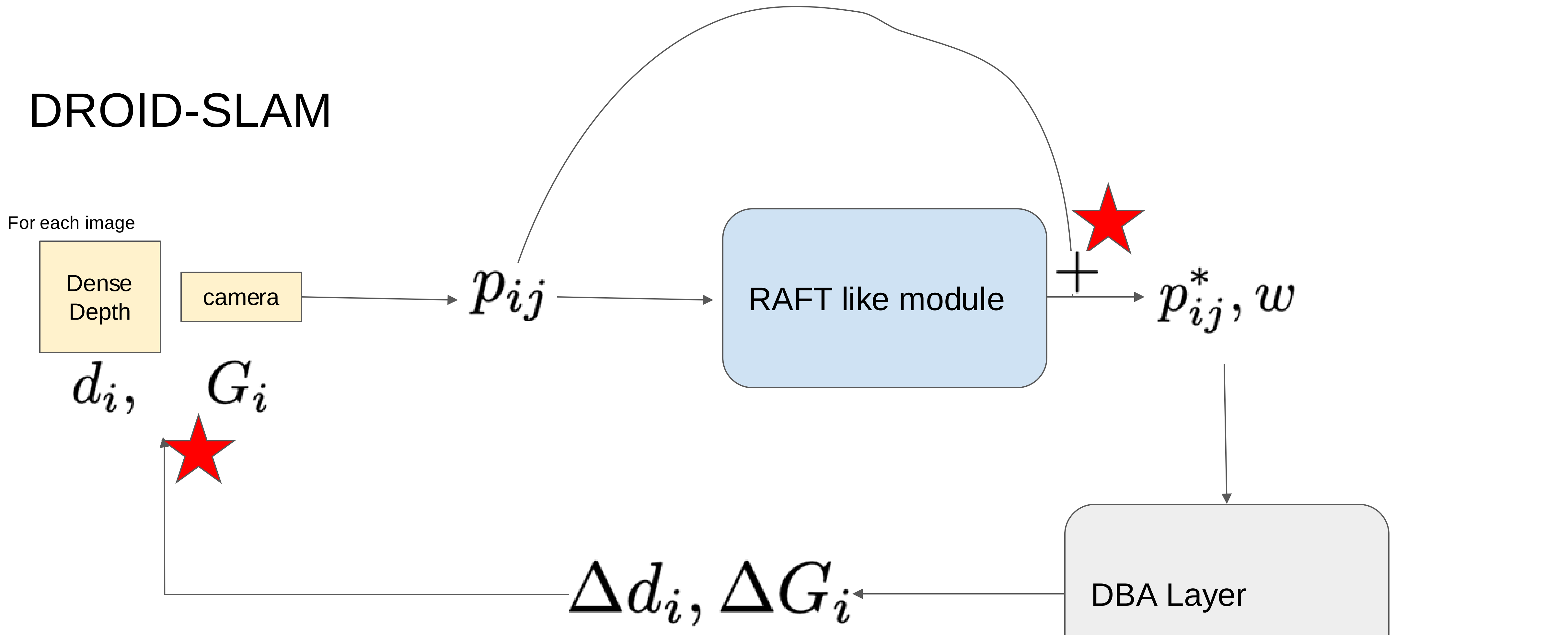
DROID-SLAM Teed and Deng, NeurIPS 2021

- Input: set of images {I_t}
- Output: for each image I_t, camera pose **G_t**, disparity **d_t**

Concept 2: Final output is not from NN but optimized


Supervision (Loss)
Current Estimates
Optimization Variables (i.e. Camera, Depth)
Correspondence (noisy Flow)
RAFT like model (Concept 1)
lookup
Computed Once
Image Features corresponding to F
I.e. Correlation Features (H x W x H x W) x L=4 many scales
Features at the current estimates
Other contextual info *
Iterative Prediction of Optimization Variable (GRU)
Correction (like Del F)
Confidence W
p^*
$+$
One step of Gauss-Newton (Big grad step)
Δ Optimization Variables
Dense Bundle Adj (DBA) layer
$\min_{\text{Optimization Variables}} ||f(\text{Optimization Variables (i.e. Camera, Depth)}) - p^*||_W$

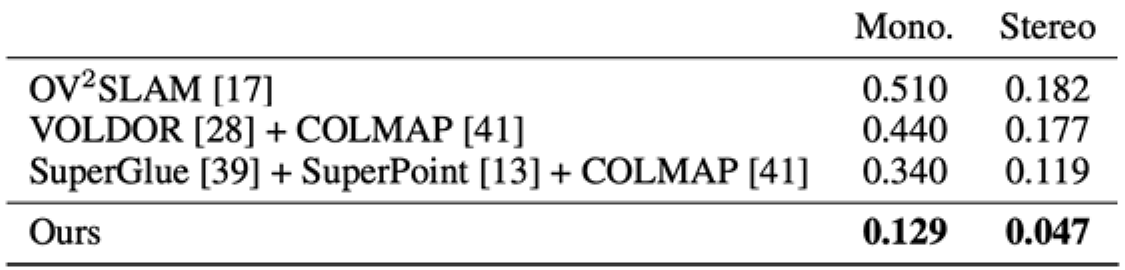
DROID-SLAM



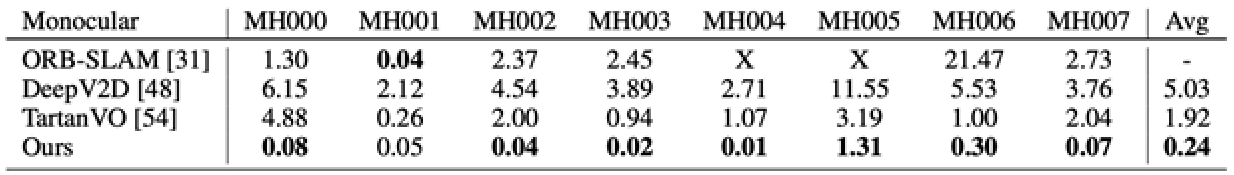
$$\mathbf{E}(\mathbf{G}', \mathbf{d}') = \sum_{(i,j)\in\mathcal{E}} \left\| \mathbf{p}^*_{ij} - \Pi_c(\mathbf{G}'_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}'_i)) \right\|^2_{\Sigma_{ij}} \qquad \Sigma_{ij} = \operatorname{diag} \mathbf{w}_{ij}.$$

Input: RGB Images
Output: Dense Depth, Camera (and flow as a consequence of these)

High accuracy: Better than COLMAP++

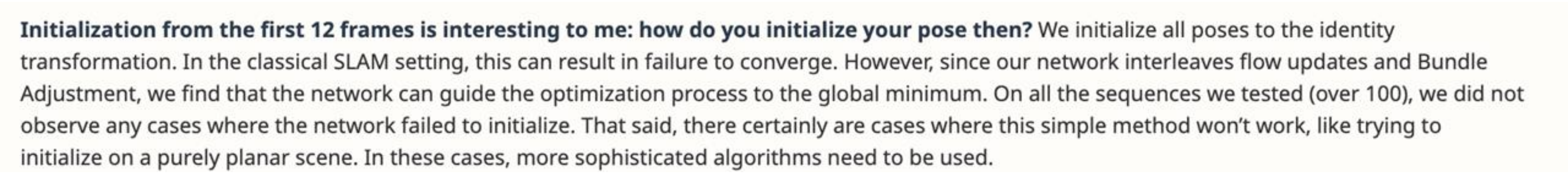


	Mono.	Stereo
OV²SLAM [17]	0.510	0.182
VOLDOR [28] + COLMAP [41]	0.440	0.177
SuperGlue [39] + SuperPoint [13] + COLMAP [41]	0.340	0.119
Ours	**0.129**	**0.047**



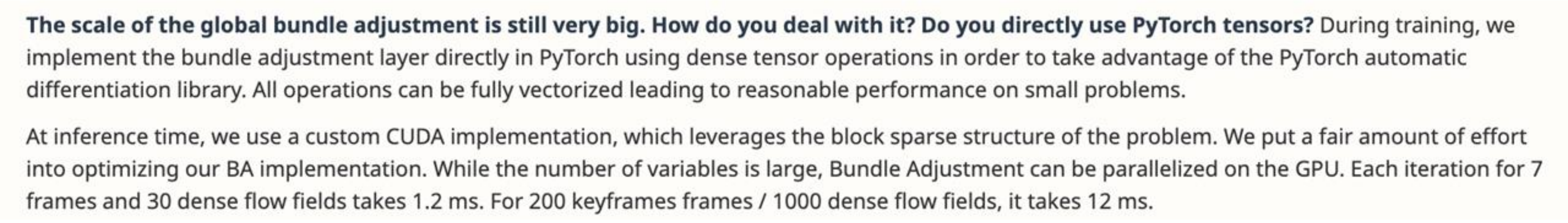
Monocular	MH000	MH001	MH002	MH003	MH004	MH005	MH006	MH007	Avg
ORB-SLAM [31]	1.30	**0.04**	2.37	2.45	X	X	21.47	2.73	-
DeepV2D [48]	6.15	2.12	4.54	3.89	2.71	11.55	5.53	3.76	5.03
TartanVO [54]	4.88	0.26	2.00	0.94	1.07	3.19	1.00	2.04	1.92
Ours	**0.08**	0.05	**0.04**	**0.02**	**0.01**	**1.31**	**0.30**	**0.07**	**0.24**

Robust:



Initialization from the first 12 frames is interesting to me: how do you initialize your pose then? We initialize all poses to the identity transformation. In the classical SLAM setting, this can result in failure to converge. However, since our network interleaves flow updates and Bundle Adjustment, we find that the network can guide the optimization process to the global minimum. On all the sequences we tested (over 100), we did not observe any cases where the network failed to initialize. That said, there certainly are cases where this simple method won't work, like trying to initialize on a purely planar scene. In these cases, more sophisticated algorithms need to be used.

Fast:



The scale of the global bundle adjustment is still very big. How do you deal with it? Do you directly use PyTorch tensors? During training, we implement the bundle adjustment layer directly in PyTorch using dense tensor operations in order to take advantage of the PyTorch automatic differentiation library. All operations can be fully vectorized leading to reasonable performance on small problems.

At inference time, we use a custom CUDA implementation, which leverages the block sparse structure of the problem. We put a fair amount of effort into optimizing our BA implementation. While the number of variables is large, Bundle Adjustment can be parallelized on the GPU. Each iteration for 7 frames and 30 dense flow fields takes 1.2 ms. For 200 keyframes frames / 1000 dense flow fields, it takes 12 ms.

Questions / Discussions

What is the learnable component?

Why doesn’t the W go to 0 (and get to a degenerate solution with everything Id, 0)?

What takes advantage of learning / semantics (deep learning prior) ?

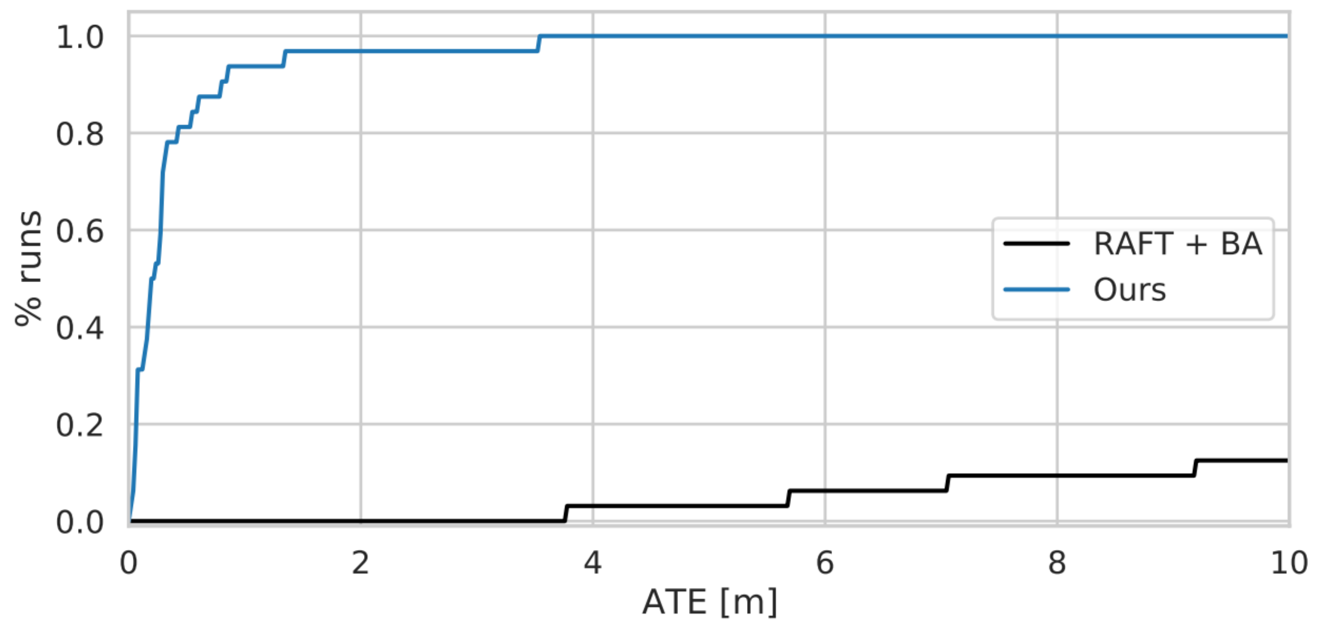
Why does this generalize (TartanAir to Tanks and Temples)?

What’s wrong if we fixed the flow from off the shelf RAFT?

Ablation: just do BA from RAFT corresp

This isn’t really clear… what happens to the confidence weights W if you just use RAFT? that can affect things a lot.

Better ablation: fix RAFT, train network that predicts W using the same context features and still do iteration



RAFT3D

Input: RGB-**D** image pair

Output: H x W x |SE(3)|, per-pixel rigid body motion

&

soft grouping (per-pixel embedding)

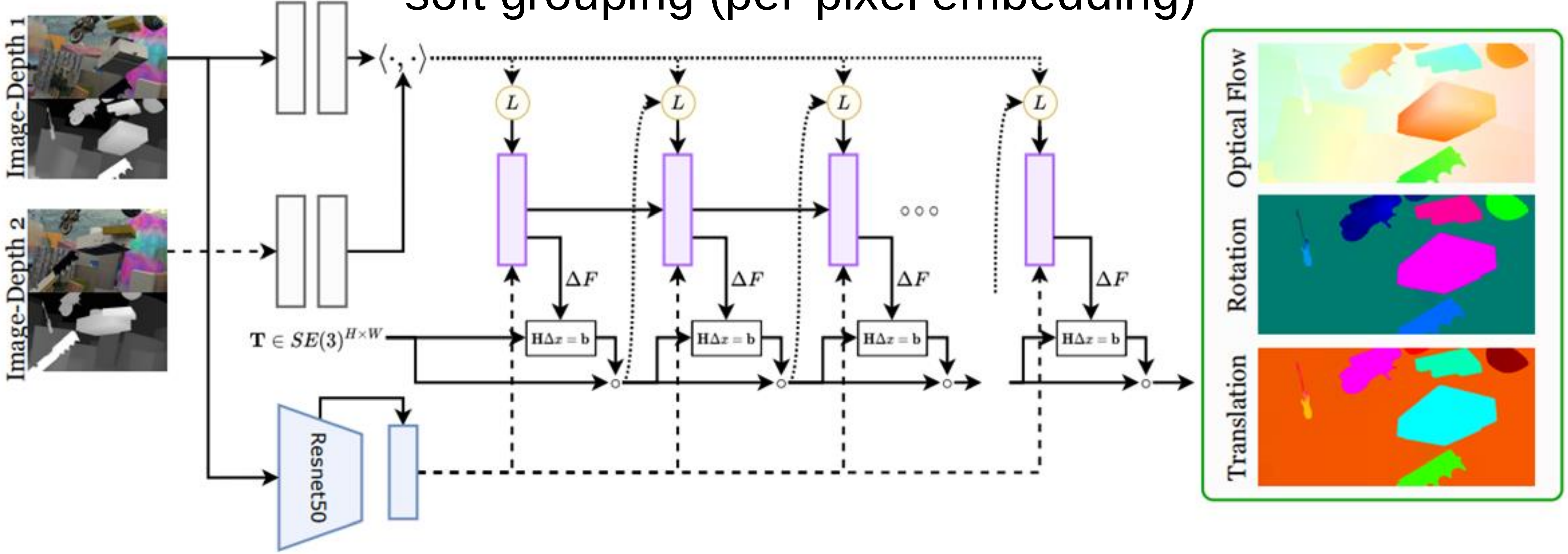


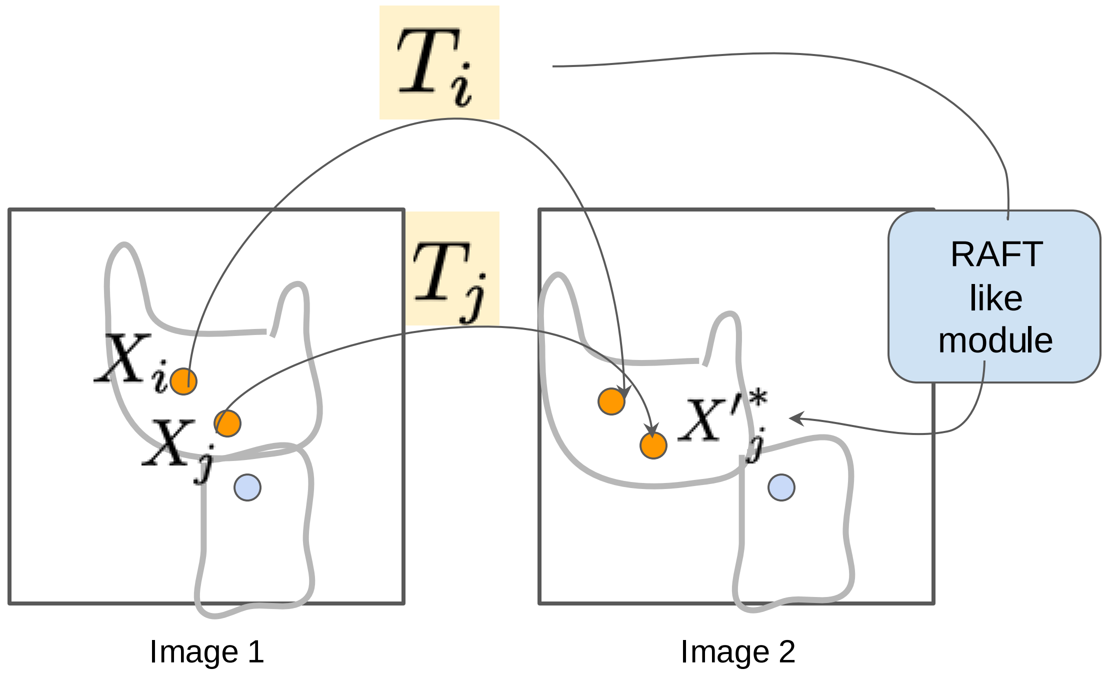
Image-Depth 1
Image-Depth 2
$\langle\cdot,\cdot\rangle$
L
ΔF
$\mathbf{H}\Delta x = \mathbf{b}$
$\mathbf{T} \in SE(3)^{H\times W}$
Resnet50
Optical Flow
Rotation
Translation

RAFT3D

Task:

- Learn per-pixel embedding to softly group
- optimization variable: per-pixel SE(3) transformations

- With objective:



$$\min_{\Delta T_j} a_{ij} || X'^*_j - \Delta T_i T_i X_j ||^2_{w_j}$$

“Move **j** by **i**’s transform, it should still go to where it went if same group”

Casual Monocular Video

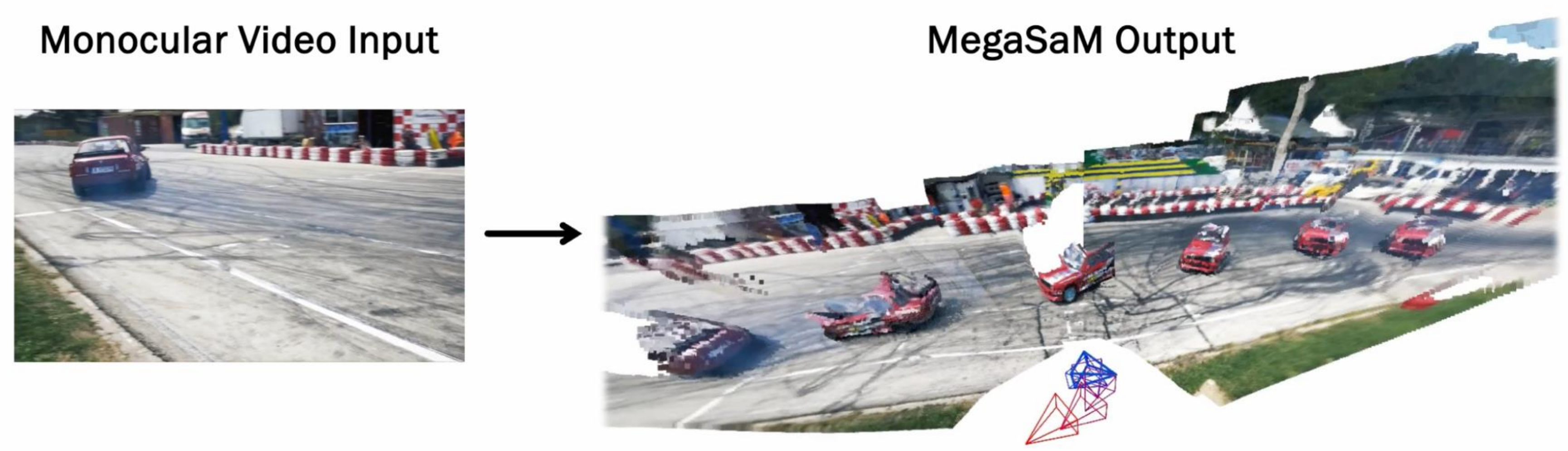
Latest Update

MegaSaM

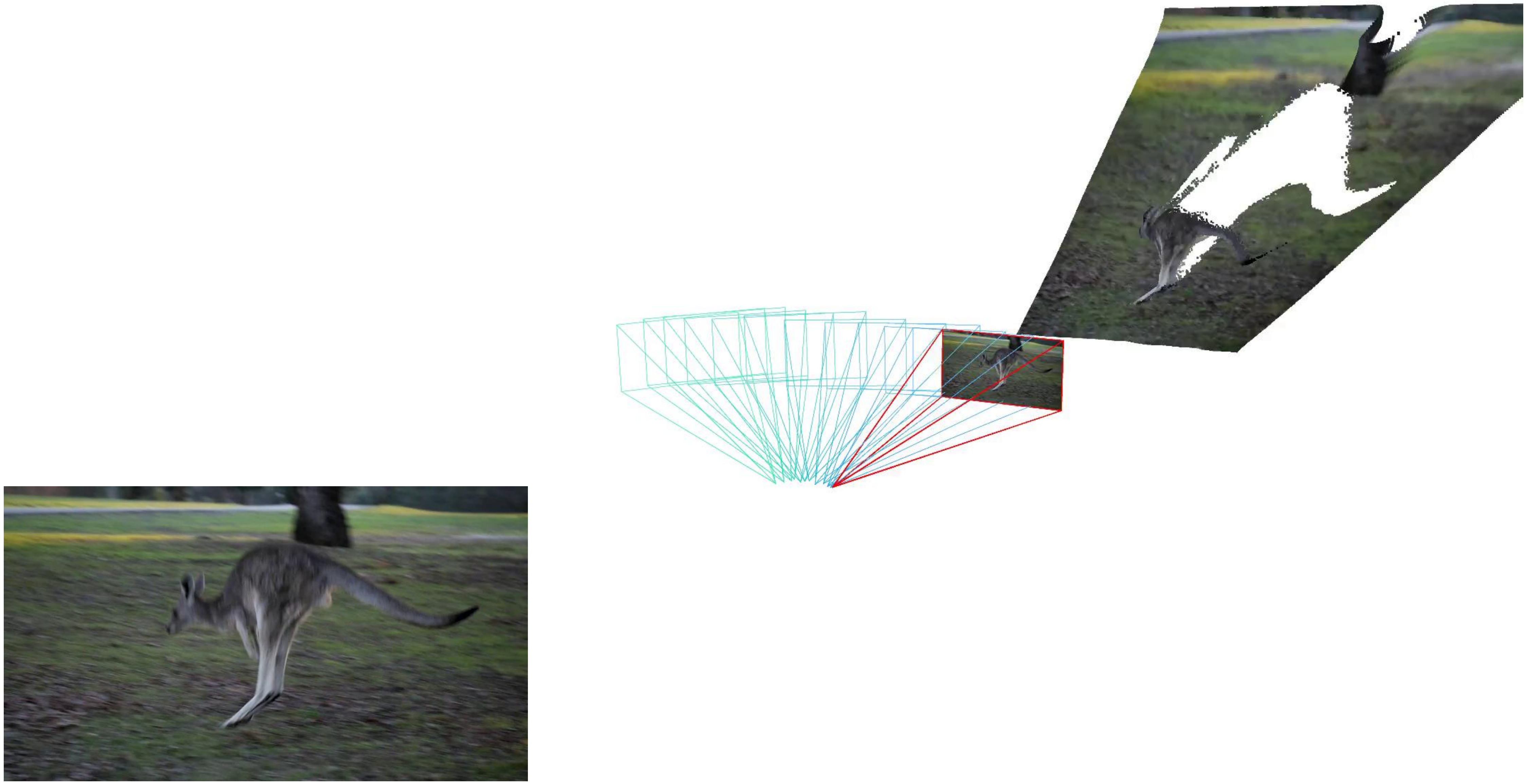
Accurate, Fast and Robust Structure and Motion from Casual Dynamic Videos

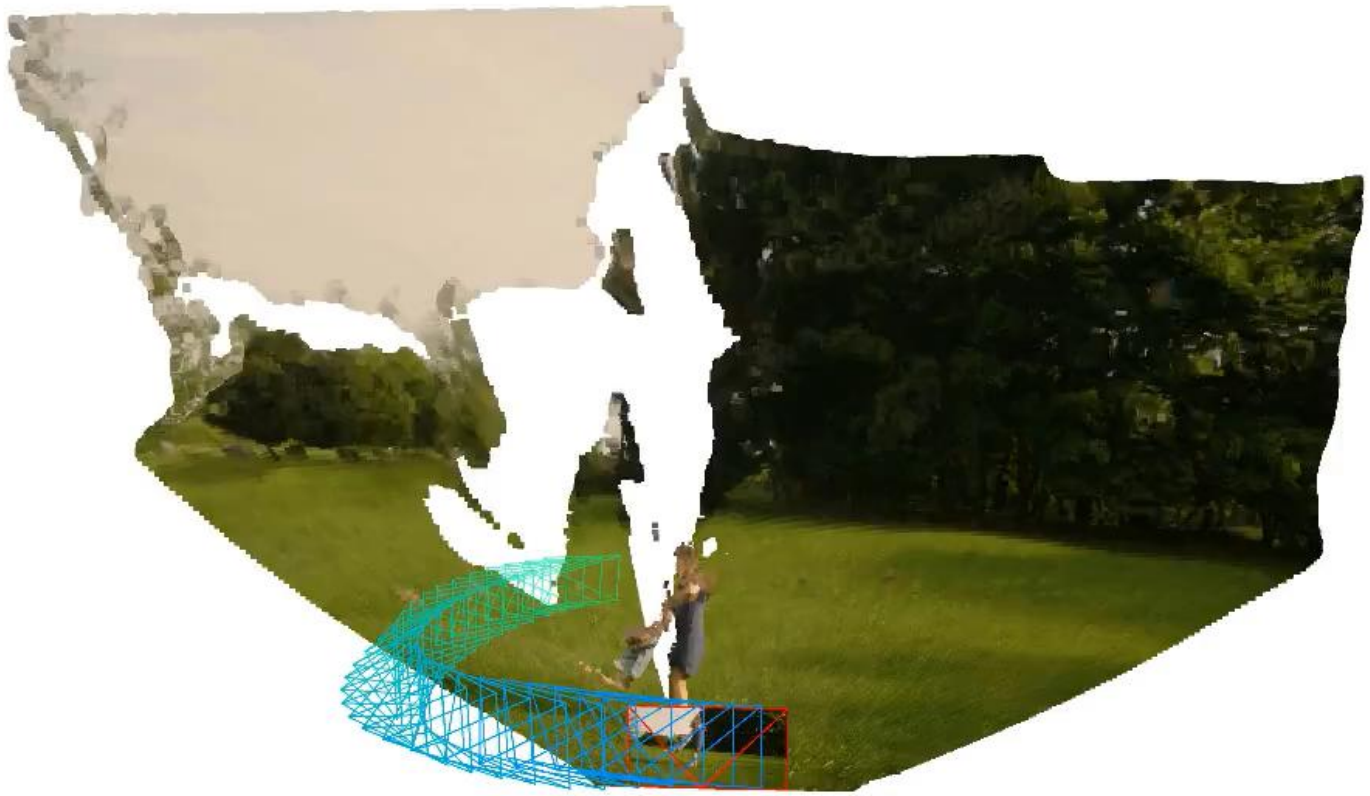
Zhengqi Li[1] Richard Tucker[1] Forrester Cole [1] Qianqian Wang[1,2] Linyi Jin[1,3] Vickie Ye[2]
Angjoo Kanazawa[2] Aleksander Holynski[1,2] Noah Snavely[1]
[1]Google DeepMind [2]UC Berkeley [3]University of Michigan

arXiv Interactive Results Gallery Code



TL;DR: MegaSaM estimates cameras and dense structure, quickly and accurately, from any static or dynamic video.





MegaSam Changes

- Optimize focal length
- Learned Object Movement Probability Map

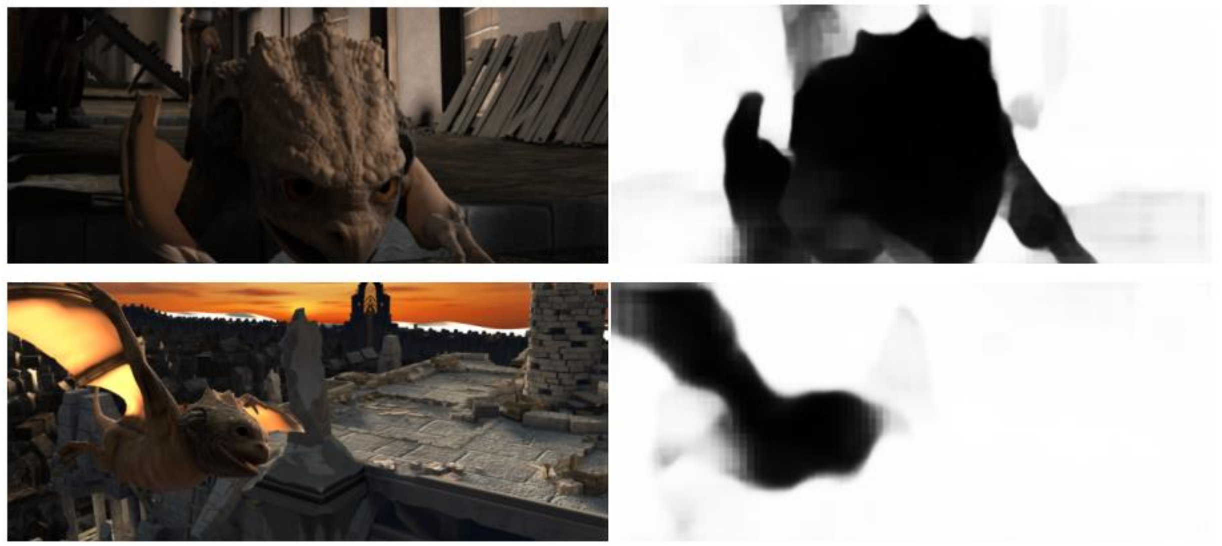
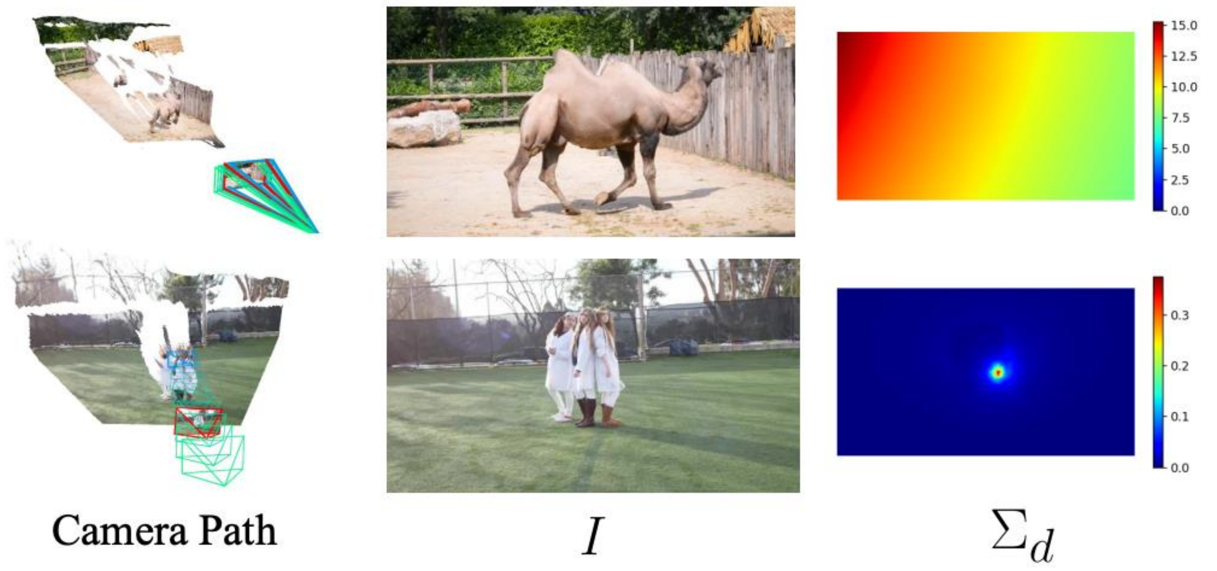


Figure 3. **Learned movement map.** Left: input video frame, right: corresponding learned motion probability map.

Changes

- Optimize focal length
- Learned Object Movement Probability Map
- Initialize D with mono-depth predictor (UniDepth)
  - In an uncertainty aware manner i.e. when to use UniDepth?
  - When camera is rotational, limited camera motion parallax



Changes

- Optimize focal length
- Learned Object Movement Probability Map
- Initialize D with mono-depth predictor (UniDepth)
  - In an uncertainty aware manner i.e. when to use UniDepth?
  - When camera is rotational, limited camera motion parallax
- Consistent video depth optimization (fix camera, flow)