ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

CORRESPONDENCE 2: POINT TRACKING, SCENE FLOW, FEATURE MATCHING



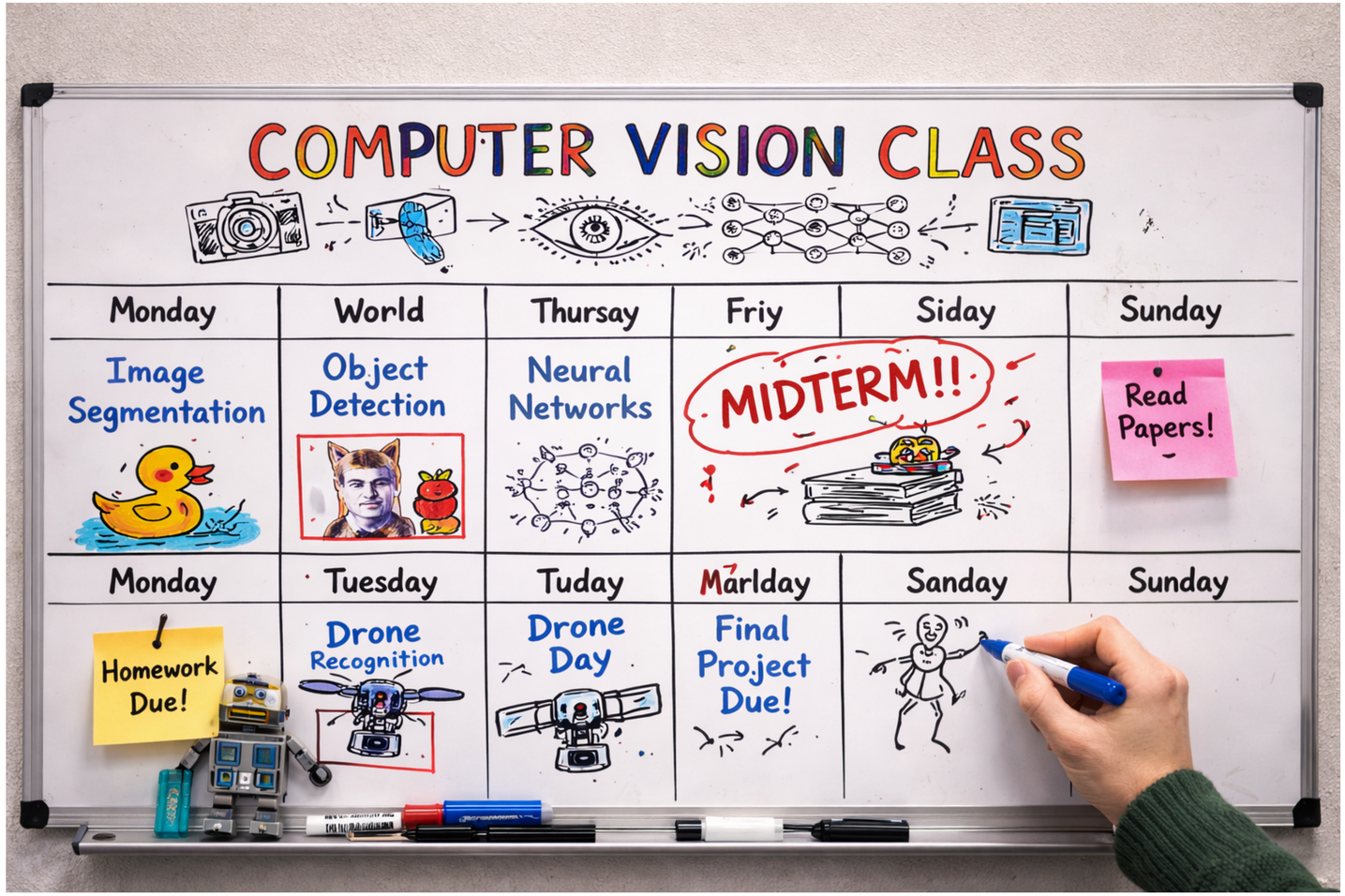
Prof. Vincent Sitzmann
Prof. Frédo Durand

Add date is Friday



Calendar updated

- Postponed vision transformers to make sure p-set dependencies work.



Mid-term

- April 9 during class. 40 minutes.
- Multiple Choice Questionnaire on paper
- All about the problem sets and corresponding lectures
  - Mostly to check you did them and understood something from them
  - Verify you understand the concepts, code, behavior
- One double-sided page of notes allowed.
  (Letter, legal or a4, whatever font size you want)



Type of questions
(on the easy side, exam will be a bit harder)

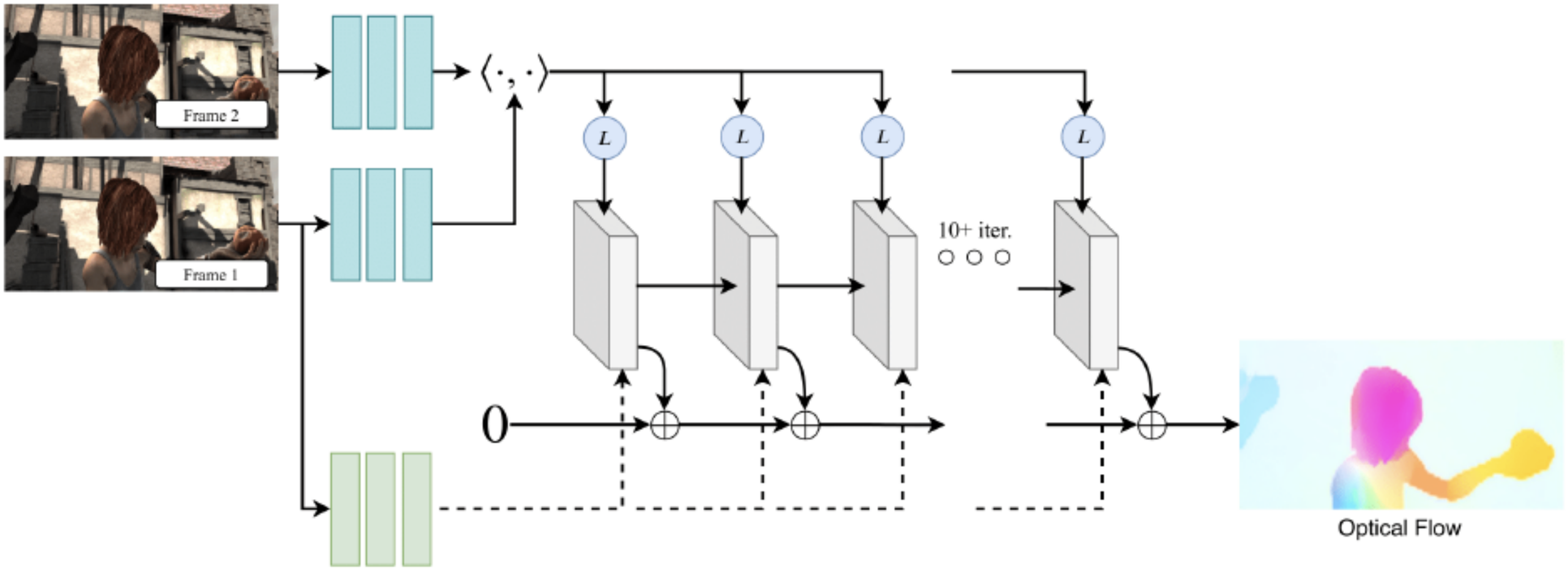
- A point (x, y, 0) in homogenous coordinates is : a point on the ground plane 2/ a direction, 3 invalid 4 the same as x, y, w for any w

- Camera intrinsics encode : 1/ the focal length 2/ the orientation 3/ the viewpoint location

- What happens in global phase-based motion magnification with a factor 2 for a scene with two points moving in opposite direction. 1/ they both move twice as fast 2/ they both move twice as far. 3/ nothing 4/ they oscillate twice as frequently

RAFT: Recurrent All-pairs Field Transforms

[Teed and Deng ECCV 2020 Best paper]



Slide credit: Teed and Deng

Another popular deep optical flow

Fast and small yet accurate

Whereas RAFT is more akin to Horn Schunk, this is more Lucas Kanade + pyramids

PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume

Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz
NVIDIA

Abstract

We present a compact but effective CNN model for optical flow, called PWC-Net. PWC-Net has been designed according to simple and well-established principles: pyramidal processing, warping, and the use of a cost volume. Cast in a learnable feature pyramid, PWC-Net uses the current optical flow estimate to warp the CNN features of the second image. It then uses the warped features and features of the first image to construct a cost volume, which is processed by a CNN to estimate the optical flow. PWC-Net is 17 times smaller in size and easier to train than the recent FlowNet2 model. Moreover, it outperforms all published optical flow methods on the MPI Sintel final pass and KITTI 2015 benchmarks, running at about 35 fps on Sintel resolution (1024 × 436) images. Our models are available on https://github.com/NVlabs/PWC-Net.

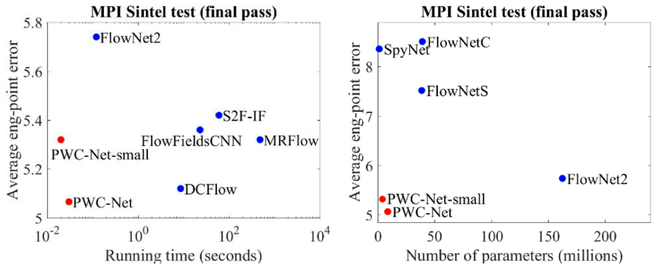
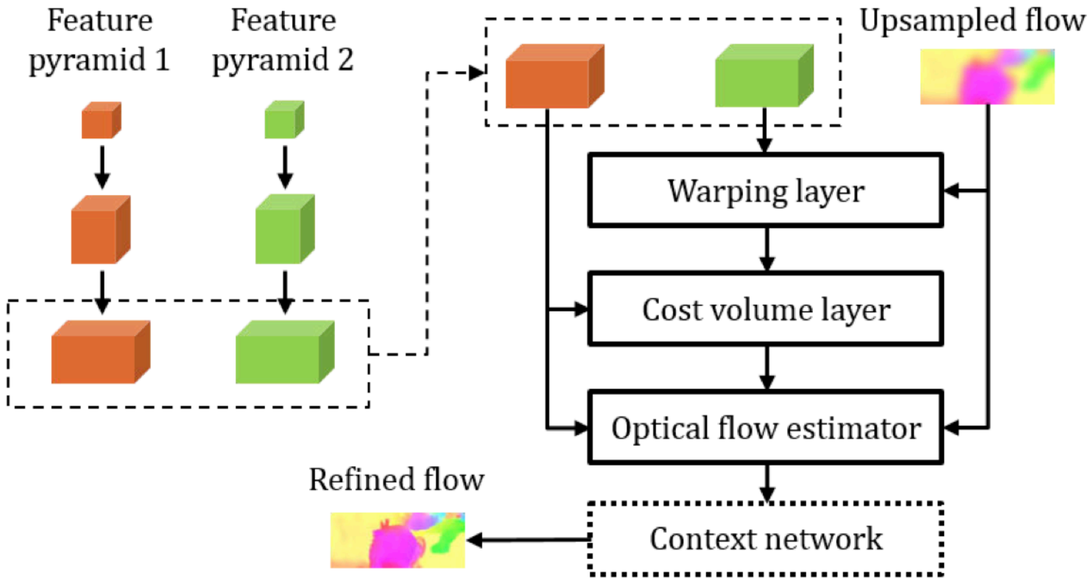


Figure 1. Left: PWC-Net outperforms all published methods on the MPI Sintel final pass benchmark in both accuracy and running time. Right: among existing end-to-end CNN models for flow, PWC-Net reaches the best balance between accuracy and size.

FlowNetS networks into a large model, called FlowNet2, which performs on par with state-of-the-art methods but runs much faster (Fig. 1). However, large models are more prone to the over-fitting problem, and as a result, the sub-networks of FlowNet2 have to be trained sequentially. Fur-

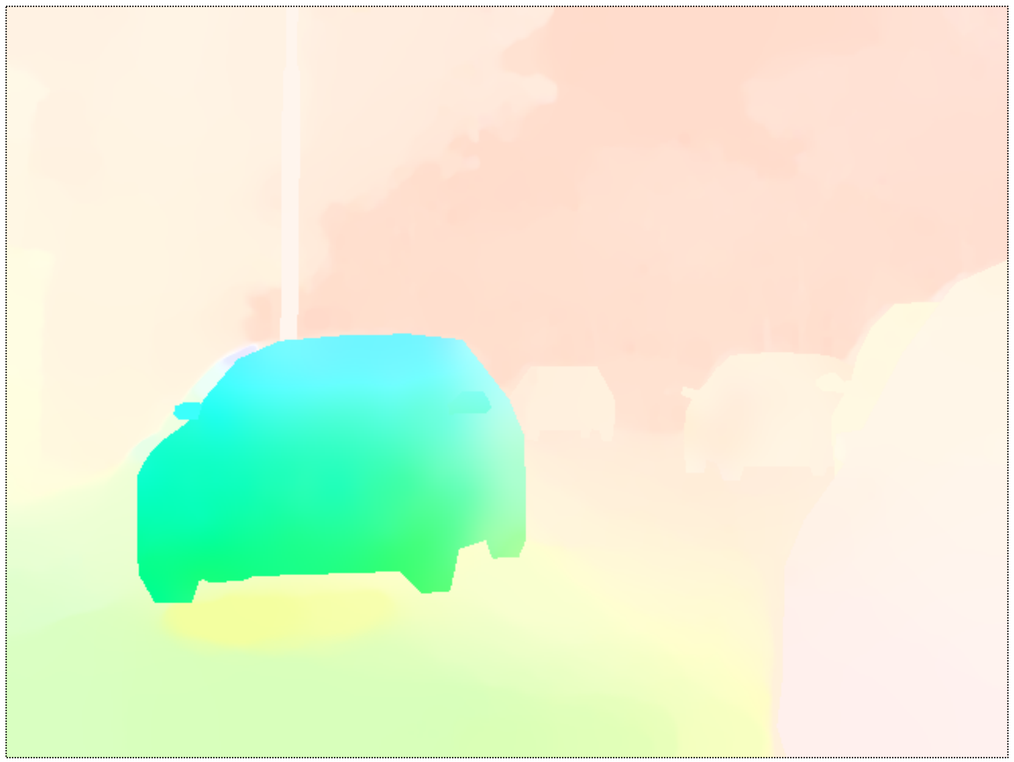


https://arxiv.org/pdf/1709.02371

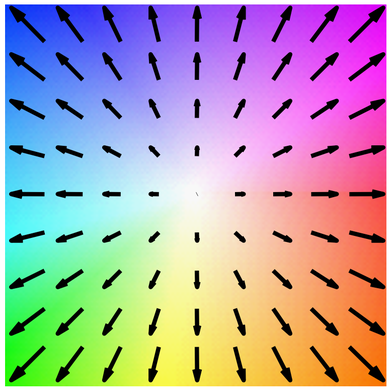
Wrapping up optical flow



Input [Liu *et al.* CVPR'08]



Optical flow (2D motion vector)



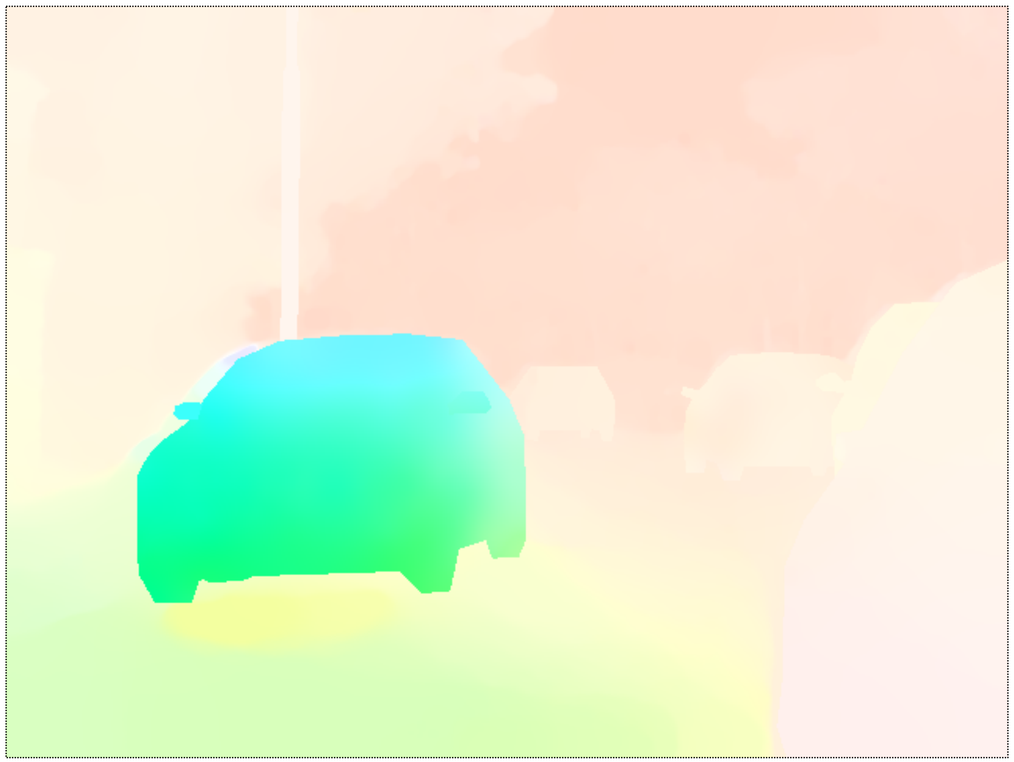
Color key
[Baker *et al.* IJCV'11]

Slide credit: Deqing Sun

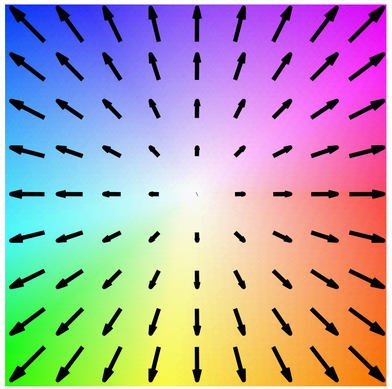
Wrapping up optical flow



Input [Liu *et al.* CVPR'08]



Optical flow (2D motion vector)

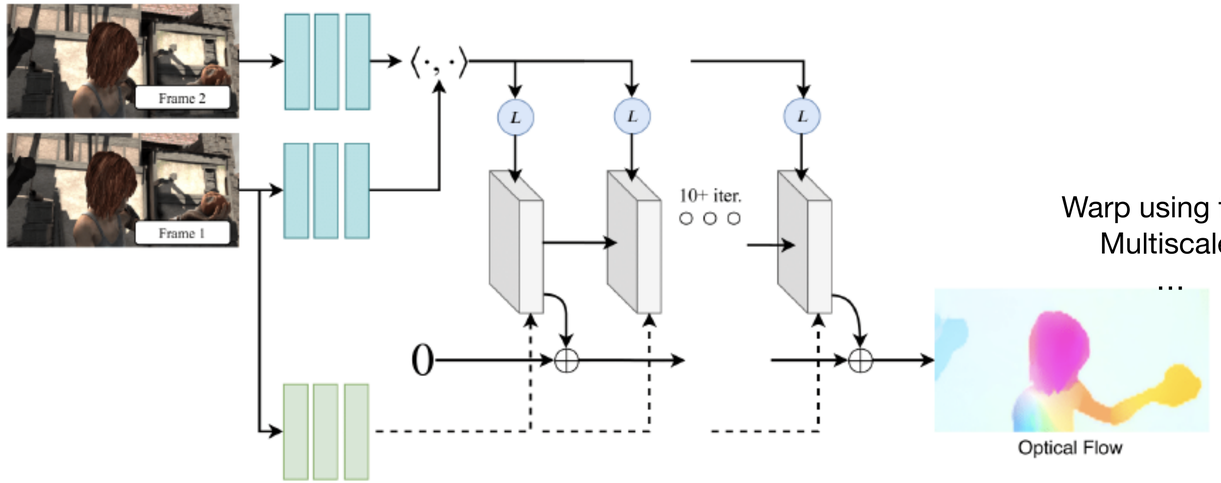


Color key
[Baker *et al.* IJCV'11]

Slide credit: Deqing Sun

Self-Supervised Optical Flow

[Stone, Maurer et al. 2021, Jonschkowski et al. 2020, Meister et al. 2018, …]
(see SMURF for good discussion!)



Warp using flow,
Multiscale,
…



$$\sum_{\mathbf{p}} \left| I_t(\mathbf{p}) - I_{t+1}\left(\mathbf{p} + \mathbf{w}_\mathbf{p}\right) \right|^2 + \lambda \sum_{\mathbf{q} \in N_p} \left| \mathbf{w}_\mathbf{p} - \mathbf{w}_\mathbf{q} \right|^2 + \dots$$

Estimate flow with Neural Network…

…but *supervise* flow with warping-based losses & regularizers!

Transformers

Transformers are good at non-local stuff

But quadratic cost makes them challenging with 4D cost volume

Requires smart encoding

PERCEIVER IO: A GENERAL ARCHITECTURE FOR STRUCTURED INPUTS & OUTPUTS

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu,

David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff,

Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, João Carreira

DeepMind

FlowFormer: A Transformer Architecture for Optical Flow

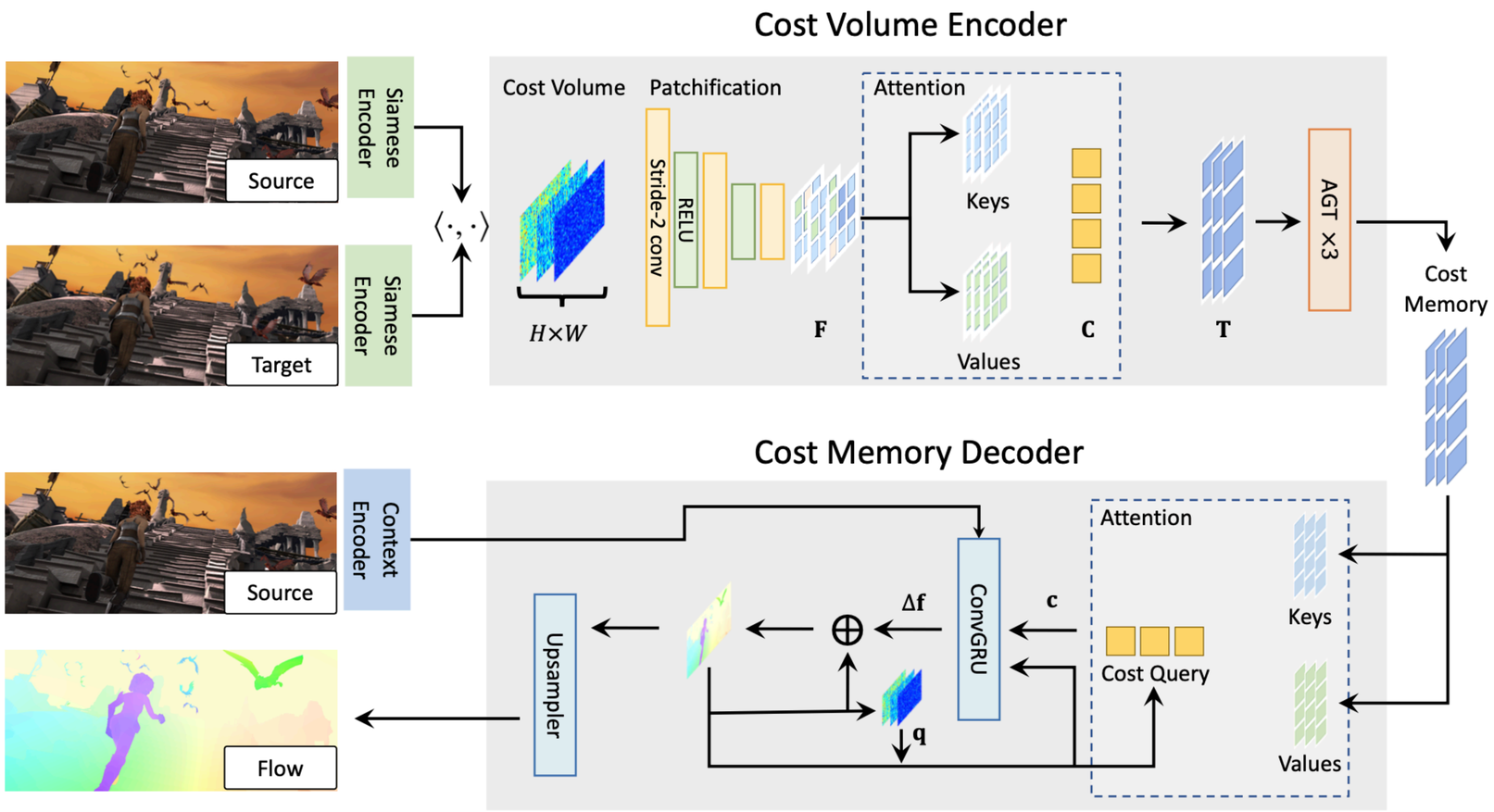
Zhaoyang Huang[1,3*], Xiaoyu Shi[1,3⋆], Chao Zhang[2], Qiang Wang[2], Ka Chun Cheung[3], Hongwei Qin[4], Jifeng Dai[4], and Hongsheng Li[1†]

FlowFormer++: Masked Cost Volume Autoencoding for Pretraining Optical Flow Estimation

Xiaoyu Shi[1,2*] Zhaoyang Huang[1,2*] Dasong Li[1] Manyuan Zhang[1]
Ka Chun Cheung[2] Simon See[2] Hongwei Qin[3] Jifeng Dai[4] Hongsheng Li[1†]

[1]Multimedia Laboratory, The Chinese University of Hong Kong
[2]NVIDIA AI Technology Center [3]SenseTime Research [4]Tsinghua University

Flowformer



Cost volume

Patchification + tokenization

-> latent 4D cost volume

Alternate-Group Transformer Layer

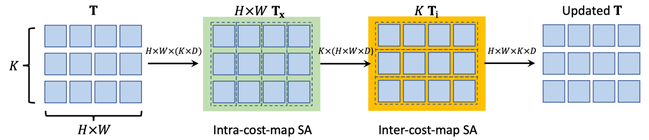


Fig. 2: Alternate-Group Transformer Layer. The alternate-group transformer layer (AGT) alternatively groups tokens in $\mathbf{T}$ into $H \times W$ groups that contains K tokens ($\mathbf{T}_x$) and K groups that contains $H \times W$ tokens ($\mathbf{T}_i$), and encode tokens inside groups via self-attention and ss self-attention [8] respectively.

Memory, self attention

Decoder still recurrent

Fig. 1: Architecture of FlowFormer. FlowFormer estimates optical flow in three steps: 1) building a 4D cost volume from image features. 2) A cost volume encoder that encodes the cost volume into the cost memory. 3) A recurrent transformer decoder that decodes the cost memory with the source image context features into flows.

FlowFormer++

Better cost volume encoding with Masked AutoEncoder

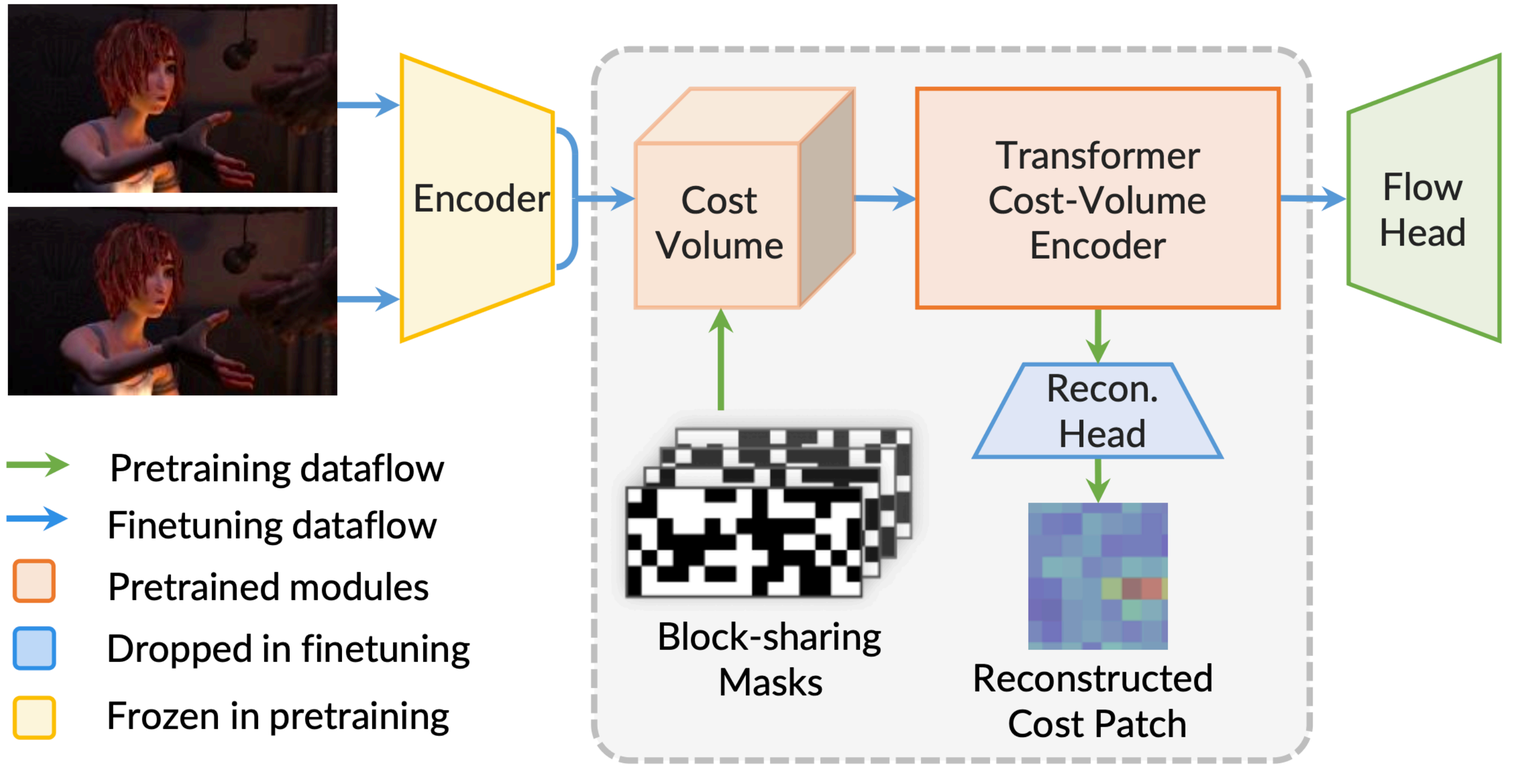
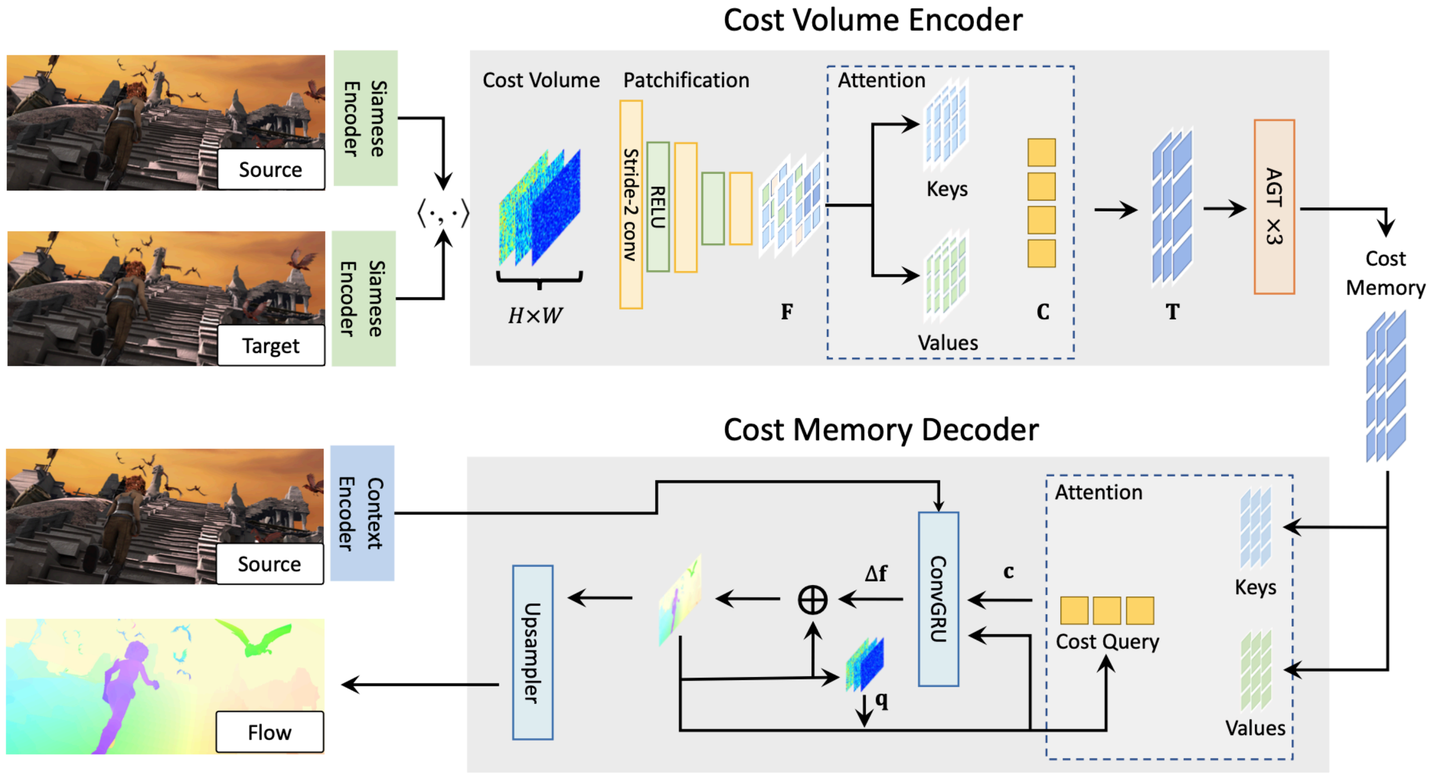


Figure 1. **Overview of FlowFormer++.** The core component of FlowFormer architecture is the transformer-based cost-volume encoder. We propose Masked Cost Volume Autoencoding to pretrain the cost-volume encoder. During pretraining, a portion of cost values are masked and the cost-volume encoder is required to reconstruct masked cost patches.

Flowformer discussion

Still specialized : cost volume, recurrence, special encoding

Transformers are great but memory is a huge challenge (especially with 4D data)



Optical Flow Discussion

Challenge: non-local matching, Lagrangian nature

- Sometimes convergence with depth estimate, sometimes not

Not always well posed (smooth areas, aperture problem, boundaries)

Has moved from manually crafted to supervised to self-supervised

Still requires some inductive biases and specialized architecture

- Not fully bitter-lessoned (yet.)

Still a fair question whether it’s a real task

- But it highlights a real type of challenge for CNNs : non-local matching

How connected is it to generative video models? Their tokenization?

Today

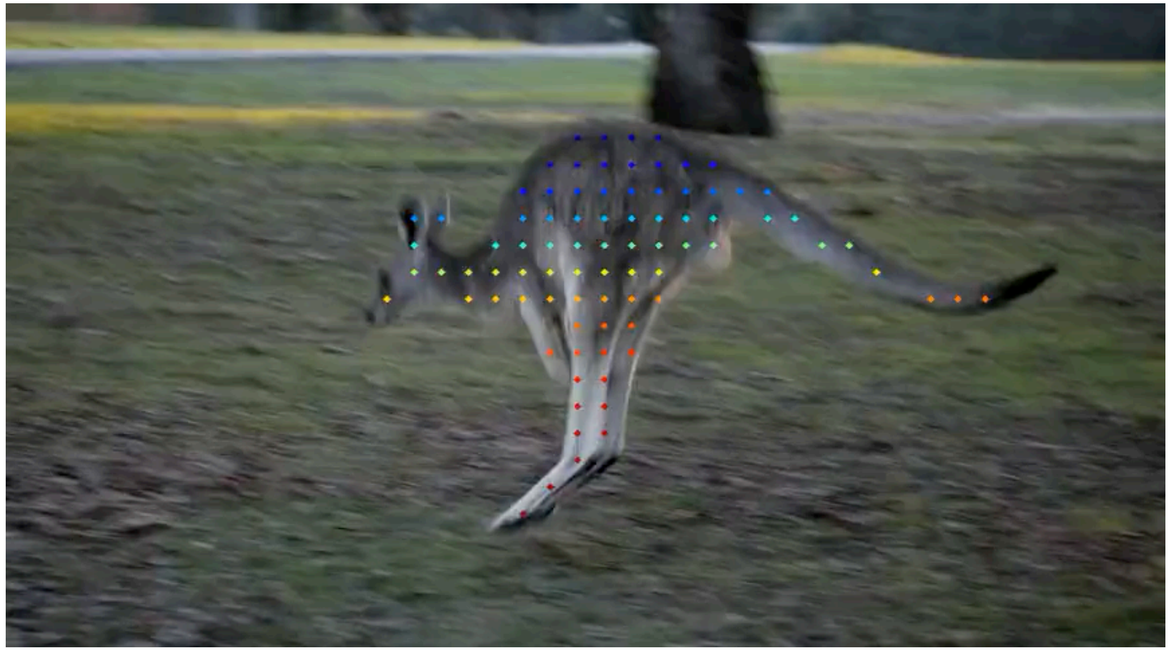
Sparse motion estimation

- Long temporal (and thus spatial) range
- Smooth over time + occlusion

Start adding 3D structure

Maybe:

- Matching between images
  (very long spatial range, no temporal smoothness)
- Ransac





https://omnimotion.github.io/

Today

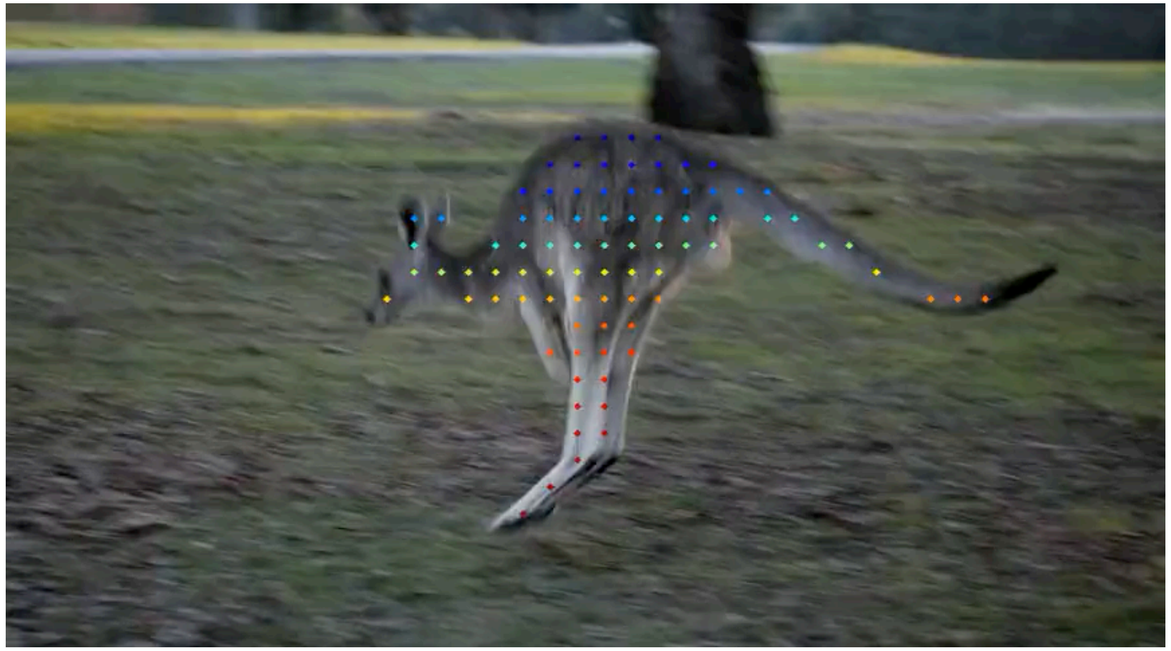
Sparse motion estimation

- Long temporal (and thus spatial) range
- Smooth over time + occlusion

Start adding 3D structure

Maybe:

- Matching between images
  (very long spatial range, no temporal smoothness)
- Ransac





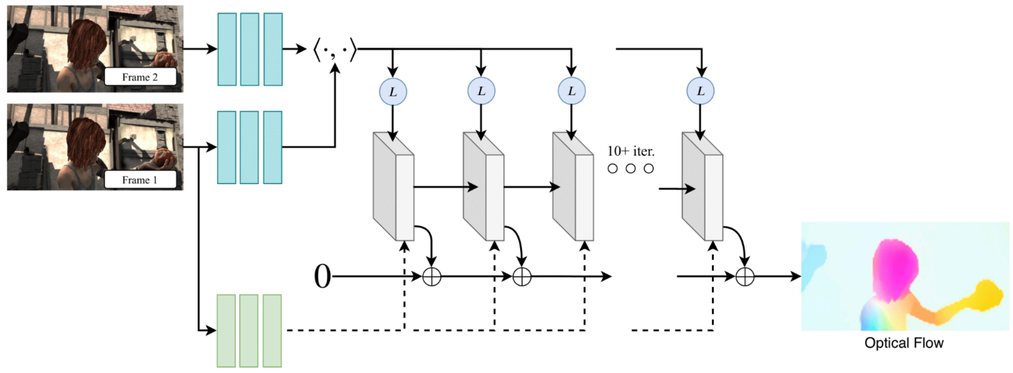
https://omnimotion.github.io/

Two approaches to Motion Estimation

Optical Flow

Dense correspondences between a pair of frames

RAFT



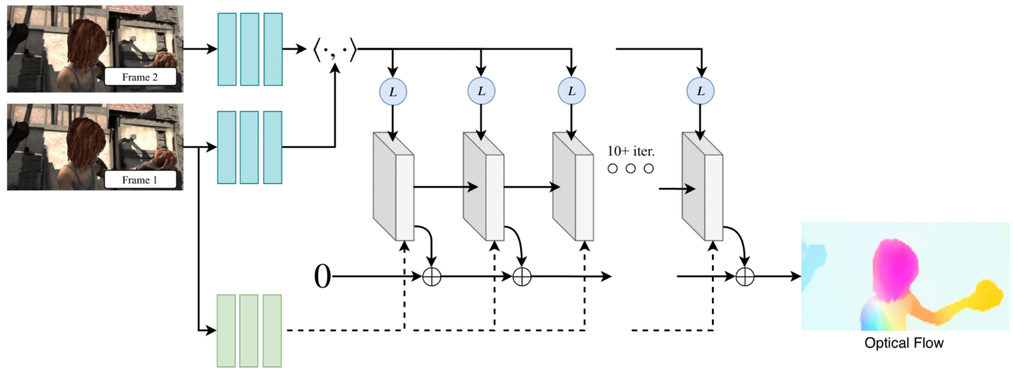
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022
TAP-Vid: A Benchmark for Tracking Any Point in a Video. Doersch et al., NeurIPS D&B 2022
RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. Teed et al. , ECCV 2020

Two approaches to Motion Estimation

Optical Flow

Dense correspondences between a pair of frames

RAFT



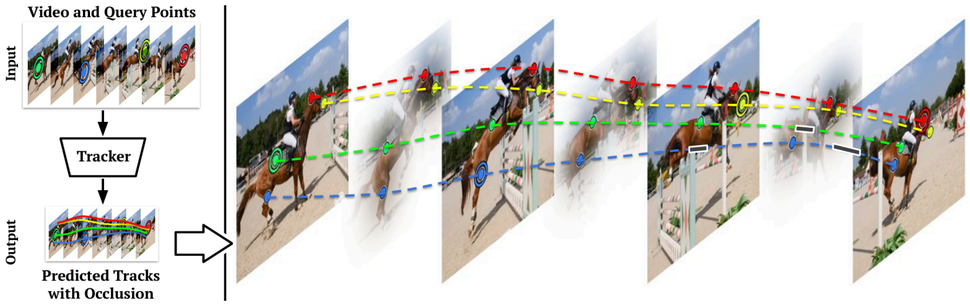
Point Tracking

Long-term tracking of individual points

PIPs



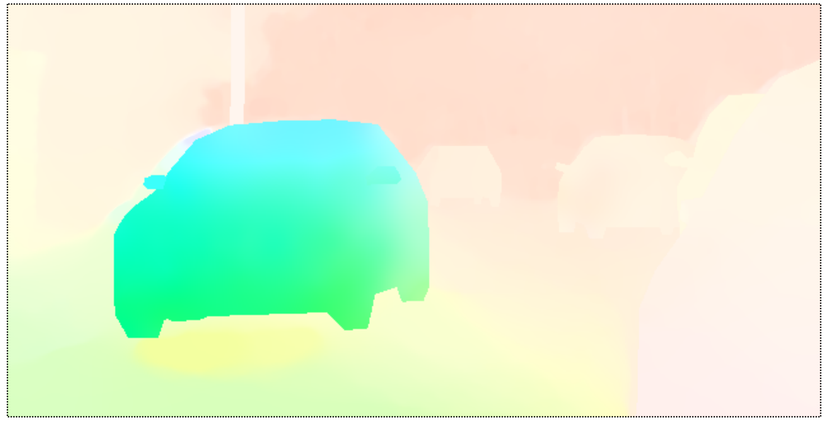
TAP-Net



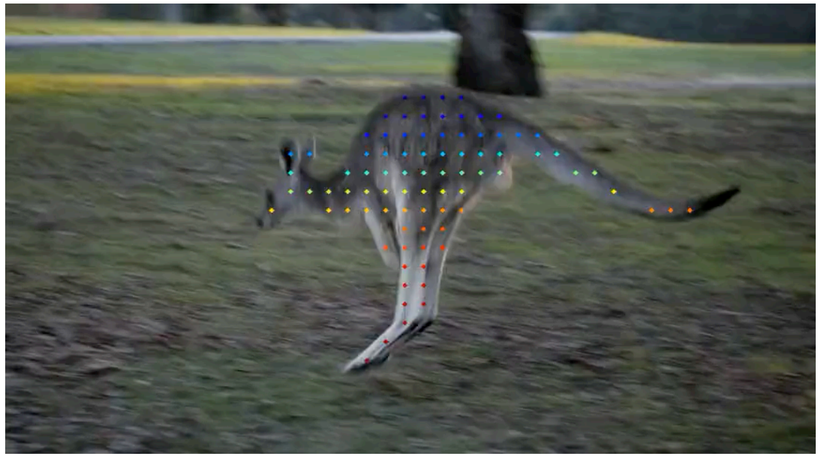
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022
TAP-Vid: A Benchmark for Tracking Any Point in a Video. Doersch et al., NeurIPS D&B 2022
RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. Teed et al. , ECCV 2020

Families of correspondence problem(s)

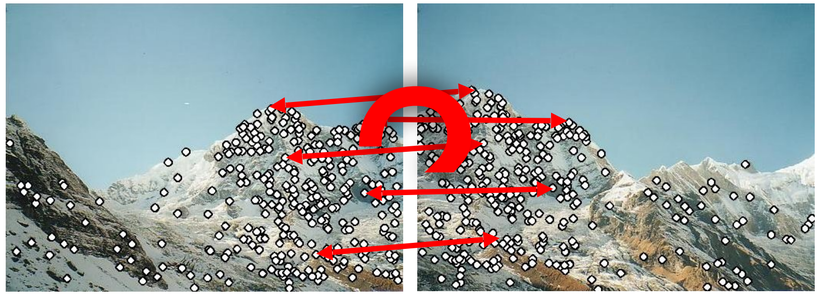
Dense (optical flow, stereo)



Sparse and smooth (+visibility)



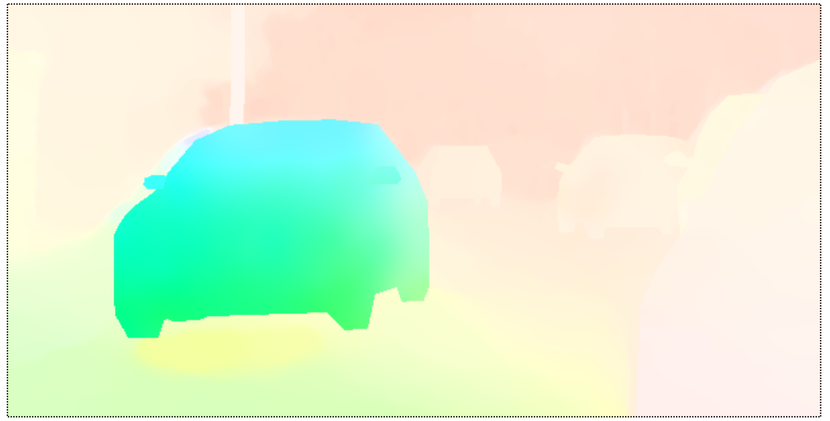
Sparse and far
(e.g. for panorama stitching, multi view stereo)



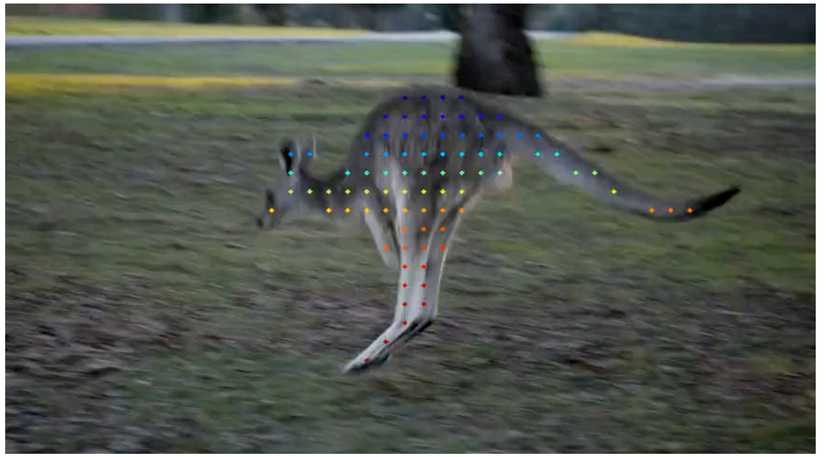
+ with or without 3D structure

Families of correspondence problem(s)

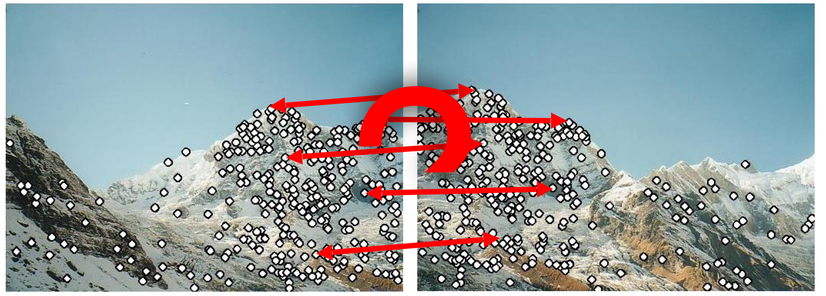
Dense (optical flow, stereo)



Sparse and smooth (+visibility)



Sparse and far
(e.g. for panorama stitching, multi view stereo)



+ with or without 3D structure

Families of correspondence problem(s)

	No 3D, motion	Static 3D
Dense	optical flow	Stereo
Sparse and smooth (+visibility)	Motion tracking	SLAM
Sparse and far	Panorama stitching	Structure From Motion

Motivations for sparse tracking

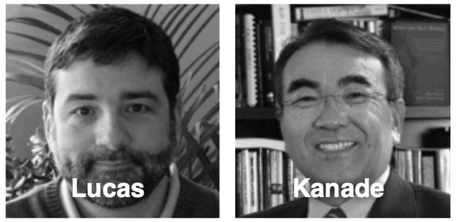
Scalability (fewer points!)

Important in the old days, still relevant today!

Robustness (better conditioning)

Some pixels are easier to track/match

History



An Iterative Image Registration Technique
with an Application to Stereo Vision.

1981

History of the

Kanade-Lucas-Tomasi (KLT) Tracker



Detection and Tracking of Feature Points.

1991

The original KLT algorithm



Good Features to Track.

1994

https://www.cs.cmu.edu/~16385/s17/Slides/15.1_Tracking__KLT.pdf

Kanade-Lucas-Tomasi

How should we track them from frame to frame?

Lucas-Kanade

Method for aligning (tracking) an image patch

How should we select features?

Tomasi-Kanade

Method for choosing the best feature (image patch) for tracking

https://www.cs.cmu.edu/~16385/s17/Slides/15.1_Tracking__KLT.pdf

Lukas Kanade optical flow

Insight: Pixels in the same patch generally share the same motion

$$\min_{\mathbf{w_p}} \sum_{\mathbf{q} \in N_p} \left(I_t(\mathbf{q}) - I_{t+1}\left(\mathbf{q} + \mathbf{w_p}\right) \right)^2$$



Image 1



Image 2

Lukas Kanade optical flow

Insight: Pixels in the same patch generally share the same motion

$$\min_{\mathbf{w_p}} \sum_{\mathbf{q} \in N_p} \left(I_t(\mathbf{q}) - I_{t+1}\left(\mathbf{q} + \mathbf{w_p}\right) \right)^2$$



Image 1



Image 2

Lukas Kanade optical flow

Insight: Pixels in the same patch generally share the same motion

$$\min_{\mathbf{w_p}} \sum_{\mathbf{q} \in N_p} \left(I_t(\mathbf{q}) - I_{t+1}\left(\mathbf{q} + \mathbf{w_p}\right) \right)^2$$



Image 1



Image 2

Lukas Kanade optical flow

$$\min_{\mathbf{w_p}} \sum_{\mathbf{q} \in N_p} \left(I_t(\mathbf{q}) - I_{t+1}\left(\mathbf{q} + \mathbf{w_p}\right) \right)^2$$

- Per-patch quadratic/linear system
  - Essentially aggregate info from multiple pixels to 1/ resolve aperture problem 2/ regularize
  - Can be seen as per-patch 1-step Horn-Schunk with strong constant smoothness
- Assumes small motion (~1 pixel max)
- One matrix equation to solve per flow vector

From Lukas Kanade optical flow **to sparse**

- Per-patch linear system
  - Essentially aggregate info from multiple pixels to 1/ resolve aperture problem 2/ regularize
  - Can be seen as per-patch 1-step Horn-Schunk with strong constant smoothness
- Assumes small motion (~1 pixel max)
- One matrix equation to solve per flow vector
- **Central idea: some such linear systems are better conditioned than others**
  - **Focus on them! Good features to track**

Concrete example: Consider translation model

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} \qquad \frac{\mathbf{W}}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Hessian

$$H = \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]^{\top} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}\right]$$

$$= \sum_{\boldsymbol{x}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_y I_x \\ \sum_{\boldsymbol{x}} I_x I_y & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$$

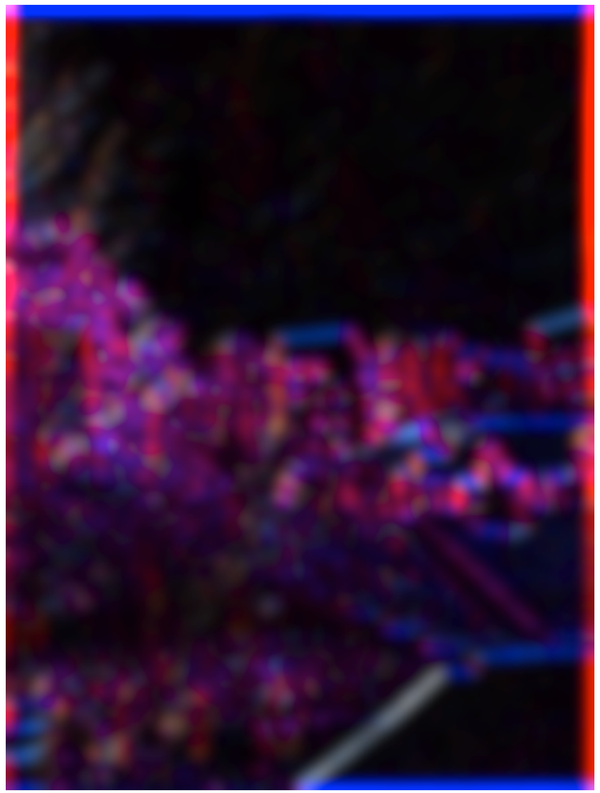
Also normal form A^tA

How are the eigenvalues related to image content?

Structure tensor

$$\begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_y I_x \\ \sum_{\boldsymbol{x}} I_x I_y & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$$





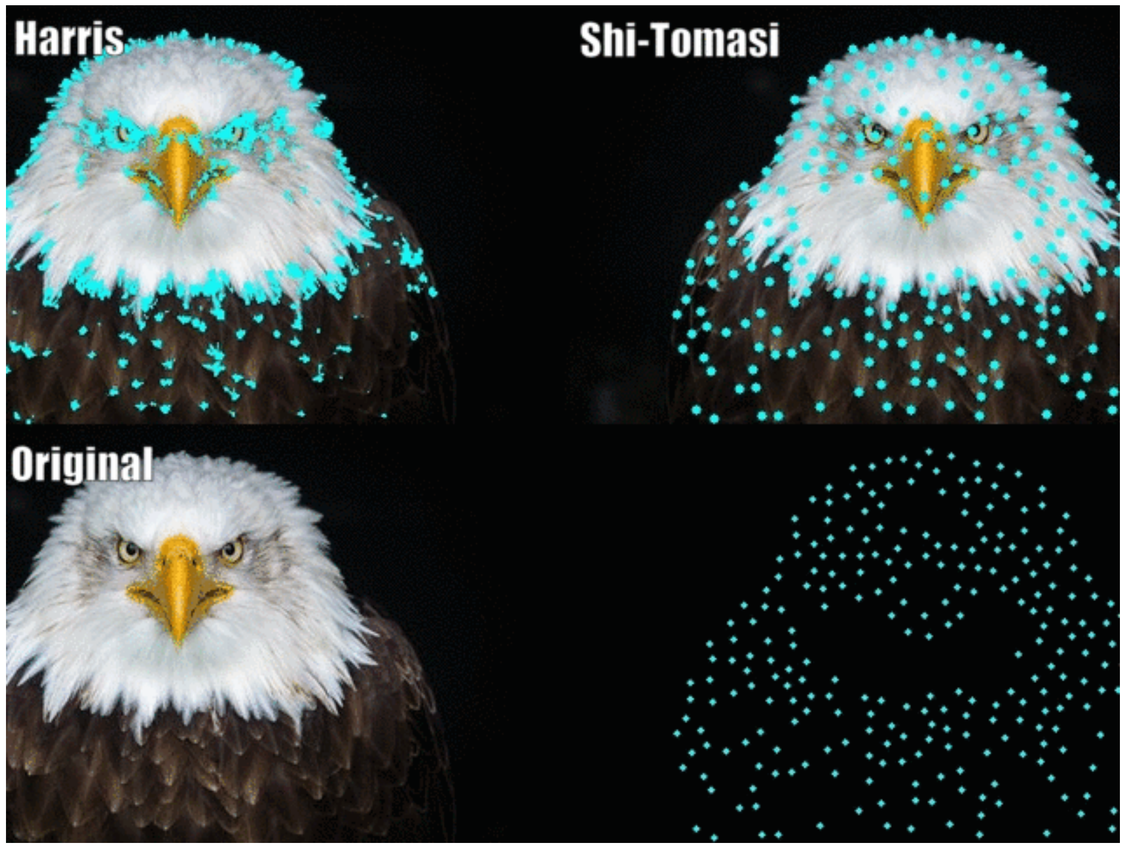
Good features and Harris corners

- Structure tensor $\begin{bmatrix} \sum_{\boldsymbol{x}} I_x I_x & \sum_{\boldsymbol{x}} I_y I_x \\ \sum_{\boldsymbol{x}} I_x I_y & \sum_{\boldsymbol{x}} I_y I_y \end{bmatrix}$
  - Captures image variation along all directions
- Eigenvalues tell us about conditioning and good sparse points to track or match
  - Shi Tomasi criterion

$$\min(\lambda_1, \lambda_2) > \lambda \ ,$$

  - Harris corner criterion

$$\det(\boldsymbol{M}) - k \cdot (\mathrm{trace}(\boldsymbol{M}))^2$$



https://medium.com/pixel-wise/detect-those-corners-aba0f034078b

Rest of the story

- Tension small point vs bigger and bigger window (full object)
- Longer temporal range
- Variation in appearance
  (because lighting changes, orientation, etc.)
- Visibility (occlusion)

Detection and tracking

Look at bigger spatial context

Combine tracking and detection

Keep refining detection model over time

Robust Object Tracking with
Online Multiple Instance Learning

Boris Babenko, *Student Member, IEEE,* Ming-Hsuan Yang, *Senior Member, IEEE*
and Serge Belongie, *Member, IEEE*

Incremental Learning for Robust Visual Tracking

David A. Ross* Jongwoo Lim† Ruei-Sung Lin‡ Ming-Hsuan Yang†
* University of Toronto † Honda Research Institute, USA ‡ Motorola Labs
dross@cs.toronto.edu jlim@honda-ri.com ruei-sung.lin@motorola.com mhyang@ieee.org

1 Aug 2015

Online Domain Adaptation for Multi-Object Tracking

Adrien Gaidon
adrien.gaidon@xrce.xerox.com
Eleonora Vig
eleonora.vig@xrce.xerox.com

Computer Vision Group
Xerox Research Centre Europe
Meylan, France

Detect to Track and Track to Detect

Christoph Feichtenhofer*
Graz University of Technology
feichtenhofer@tugraz.at

Axel Pinz
Graz University of Technology
axel.pinz@tugraz.at

Andrew Zisserman
University of Oxford
az@robots.ox.ac.uk

Robust Object Tracking with Online Multiple Instance Learning

Boris Babenko, *Student Member, IEEE,* Ming-Hsuan Yang, *Senior Member, IEEE* and Serge Belongie, *Member, IEEE*

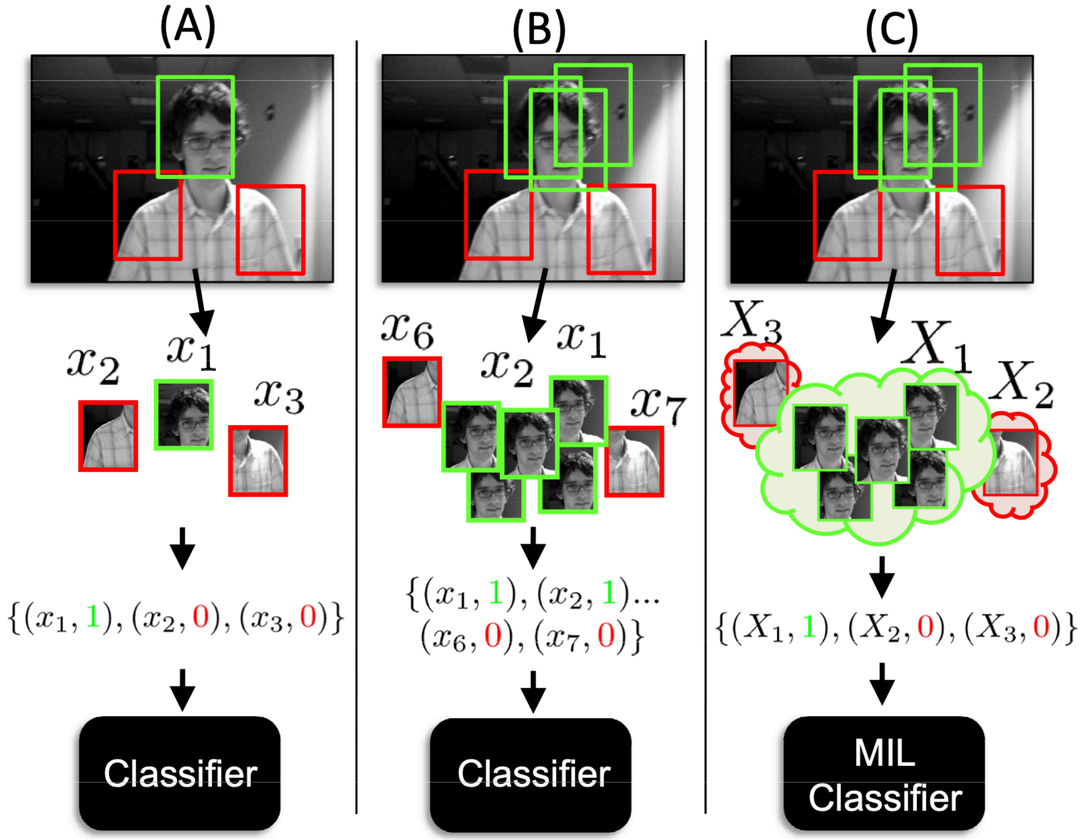


Fig. 1. **Updating a discriminative appearance model:** (A) Using a single positive image patch to update a traditional discriminative classifier. The positive image patch chosen does not capture the object perfectly. (B) Using several positive image patches to update a traditional discriminative classifier. This can make it difficult for the classifier to learn a tight decision boundary. (C) Using one positive bag consisting of several image patches to update a MIL classifier. See Section 4 for empirical results of these three strategies.

Longer time range

Reducing Drift in Parametric Motion Tracking

A. Rahimi L.-P. Morency T. Darrell

MIT AI Lab
Cambridge, MA 02139

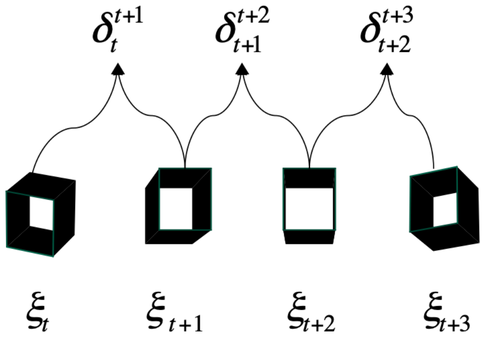


Figure 1: Independence diagram for a simple pose tracker. The tracker measures pose differences $\{\delta\}$ between adjacent frames.

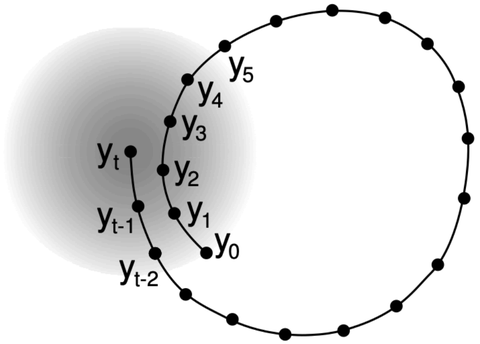


Figure 2: When estimating the pose of frame y_t, we should take into account the pose change between y_t and y_{t-1} as well as all other frames which are in the shaded region.

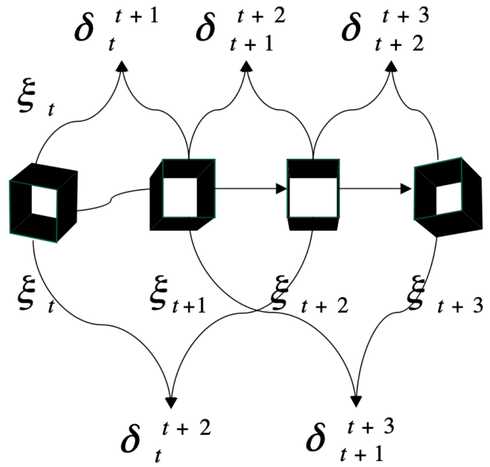


Figure 3: The measurement model when multiple base frames are used. A dynamical model for pose change is also added (horizontal arrows).

Particle Video:
Long-Range Motion Estimation using Point Trajectories

Peter Sand and Seth Teller
MIT Computer Science and Artificial Intelligence Laboratory
{sand,teller}@csail.mit.edu

Abstract

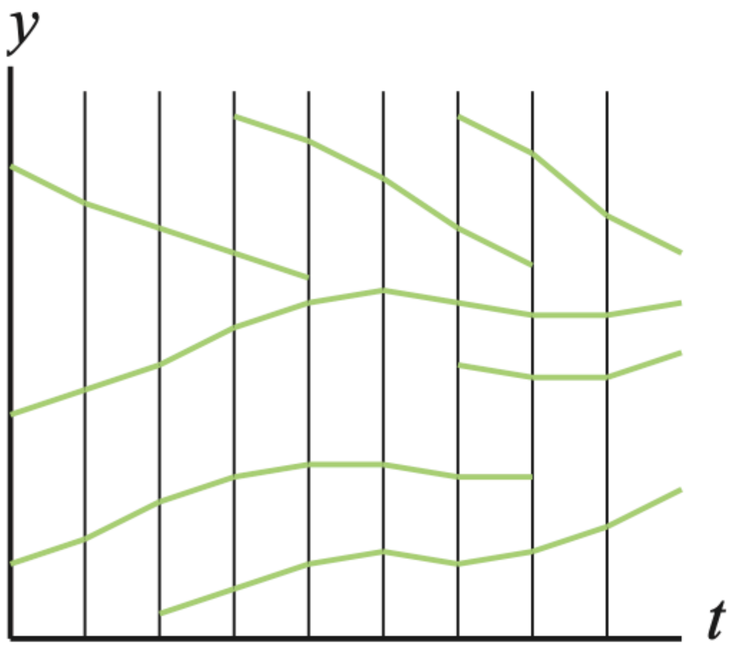
This paper describes a new approach to motion estimation in video. We represent video motion using a set of particles. Each particle is an image point sample with a long-duration trajectory and other properties. To optimize these particles, we measure point-based matching along the particle trajectories and distortion between the particles. The resulting motion representation is useful for a variety of applications and cannot be directly obtained using existing methods such as optical flow or feature tracking. We demonstrate the algorithm on challenging real-world videos that include complex scene geometry, multiple types of occlusion, regions with low texture, and non-rigid deformations.

This form of motion estimation is useful for a variety of applications. Multiple observations of each scene point can be combined for super-resolution, noise removal, segmentation, and increased effective dynamic range. The correspondences can also improve the temporal coherence of image filters that operate independently on each frame. Additionally, long-range motion estimation can simplify interactive video manipulation, including matting, rotoscoping, and object removal.

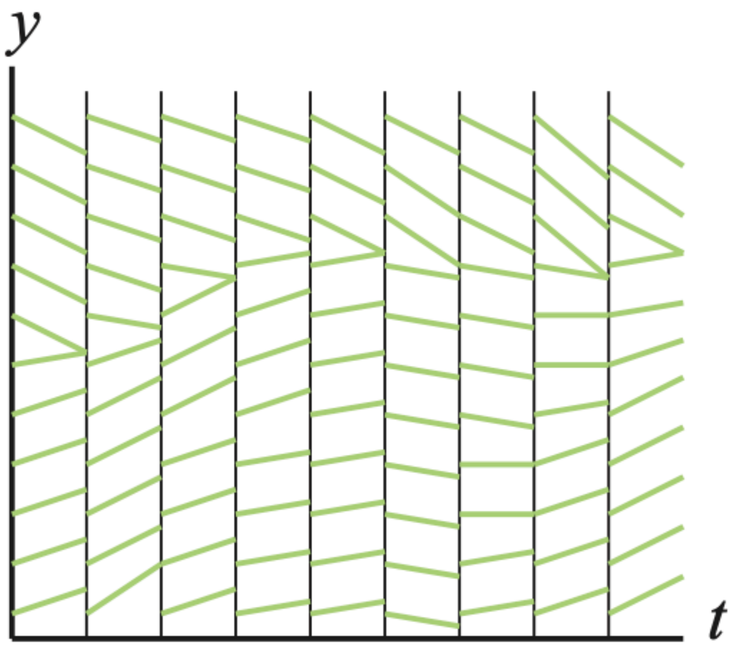
1.1. Particle Video Representation

Our approach represents video motion using a set of particles that move through time. Each particle denotes an interpolated image point sample, in contrast to a feature patch that represents a neighborhood of pixels [12]. Particle den-

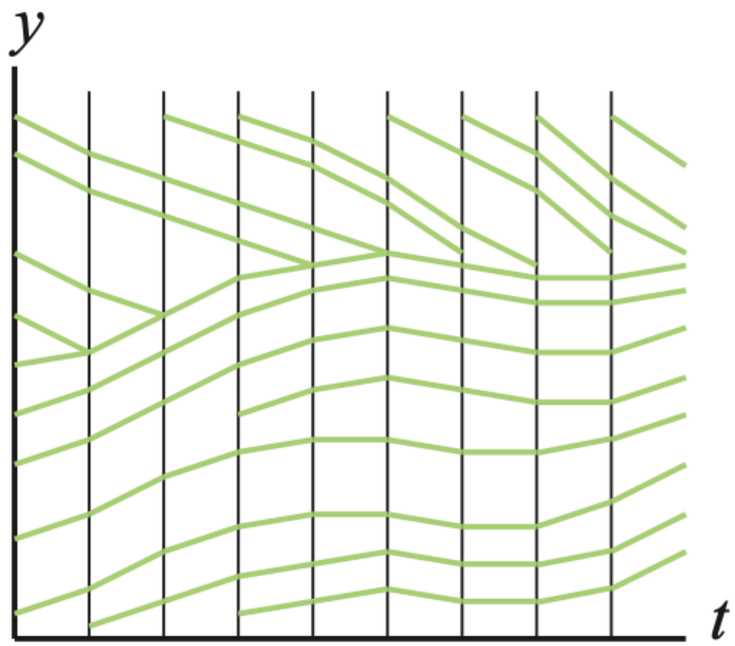
Particle videos : not fully dense, less sparse



Feature Tracking



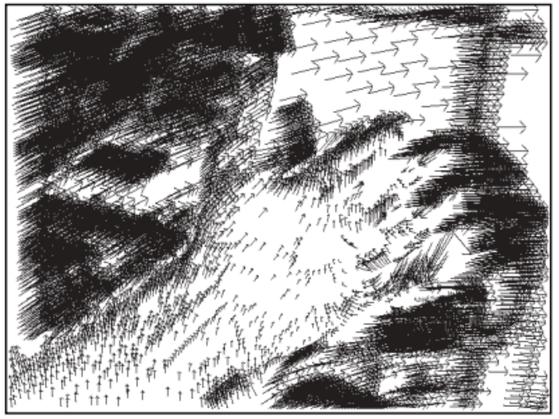
Optical Flow



Particle Video



Video 1, Frame 0



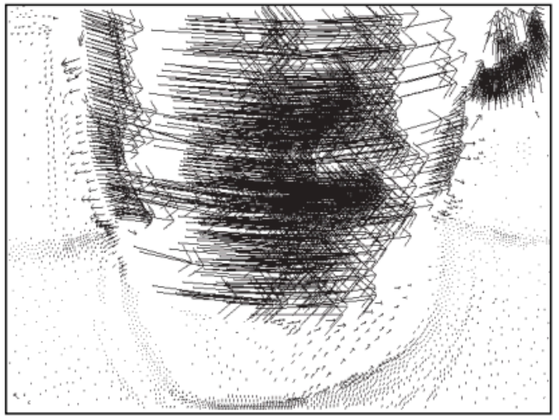
Particle Correspondences



Video 1, Frame 25



Video 2, Frame 0



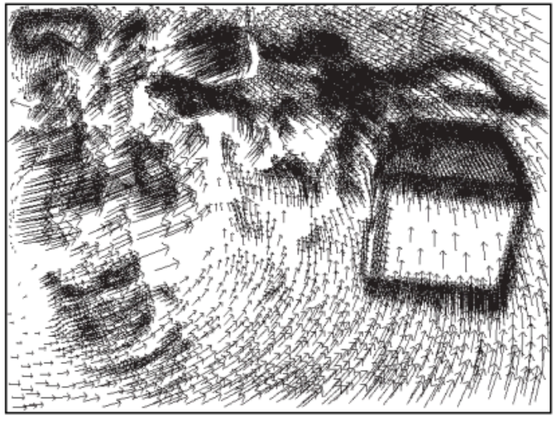
Particle Correspondences



Video 2, Frame 25



Video 3, Frame 0



Particle Correspondences



Video 3, Frame 25

Not exactly particle videos but related Sand & Teller



https://people.csail.mit.edu/teller/pubs/videomatching.pdf

Not exactly particle videos but related Sand & Teller



https://people.csail.mit.edu/teller/pubs/videomatching.pdf

2207.12153v2 [cs.CV] 25 Jul 2022

Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories

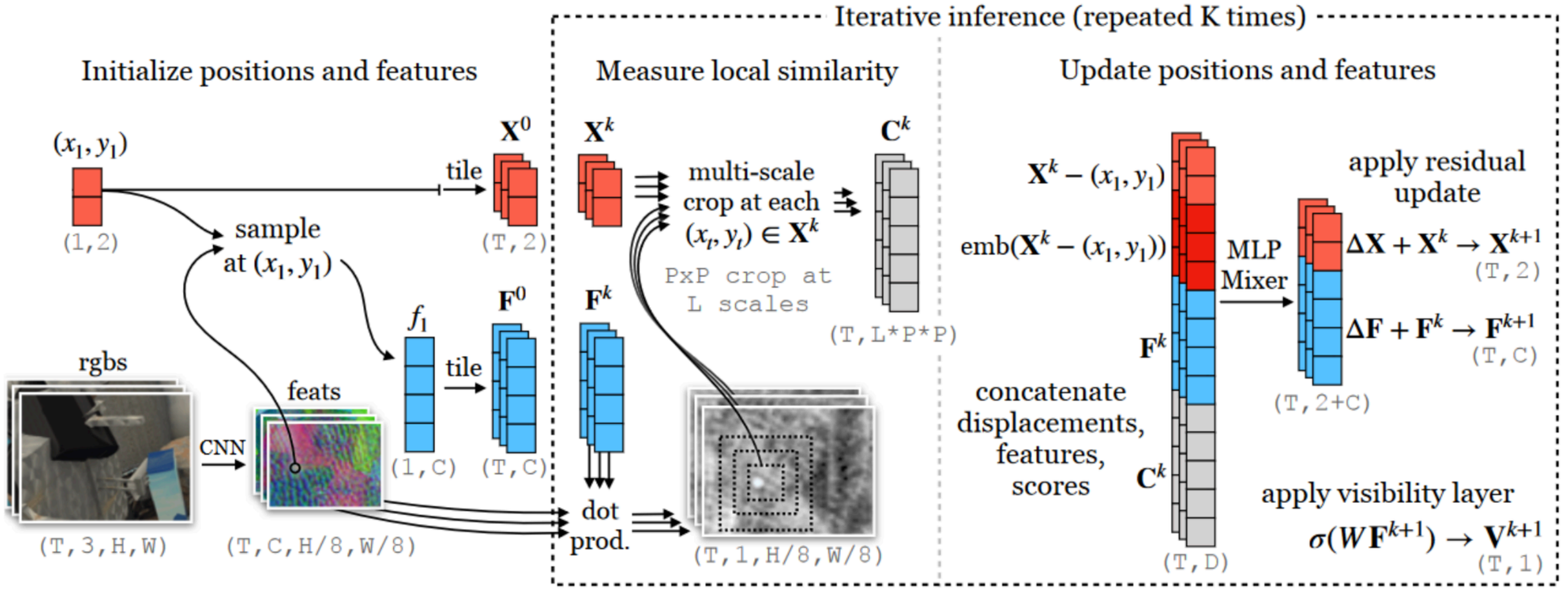
Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki

Carnegie Mellon University
{aharley, zhaoyuaf, katef}@cs.cmu.edu
Project page: https://particle-video-revisited.github.io

Abstract. Tracking pixels in videos is typically studied as an optical flow estimation problem, where every pixel is described with a displacement vector that locates it in the next frame. Even though wider temporal context is freely available, prior efforts to take this into account have yielded only small gains over 2-frame methods. In this paper, we revisit Sand and Teller's "particle video" approach, and study pixel tracking as a long-range motion estimation problem, where every pixel is described with a trajectory that locates it in multiple future frames. We re-build this classic approach using components that drive the current state-of-the-art in flow and object tracking, such as dense cost maps, iterative optimization, and learned appearance updates. We train our models using long-range amodal point trajectories mined from existing optical flow data that we synthetically augment with multi-frame occlusions. We test our approach in trajectory estimation benchmarks and in keypoint label propagation tasks, and compare favorably against state-of-the-art optical flow and feature tracking methods.

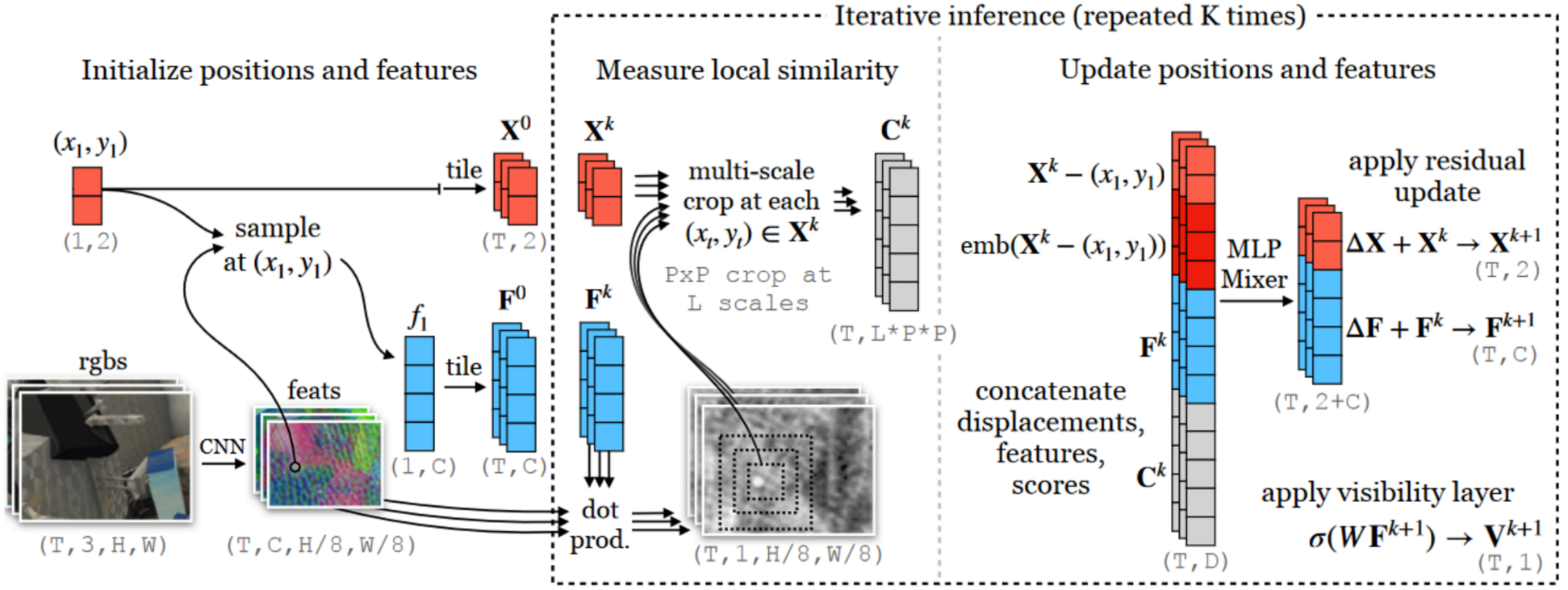
PIP

- Supervised on synthetic data
- Based on RAFT
- Tracks single point at a time (for tractability reasons)



PIP

- Iterative solution
  - Solve for x,y + appearance vector for all frame
  - Predict whether point visible in each frame
  - Initialize with constant location and appearance



Particle Video Revisited

User input

Initialize positions and features

(x_1, y_1)

(1,2)

1 based indexing for some reason ;-(

rgbs

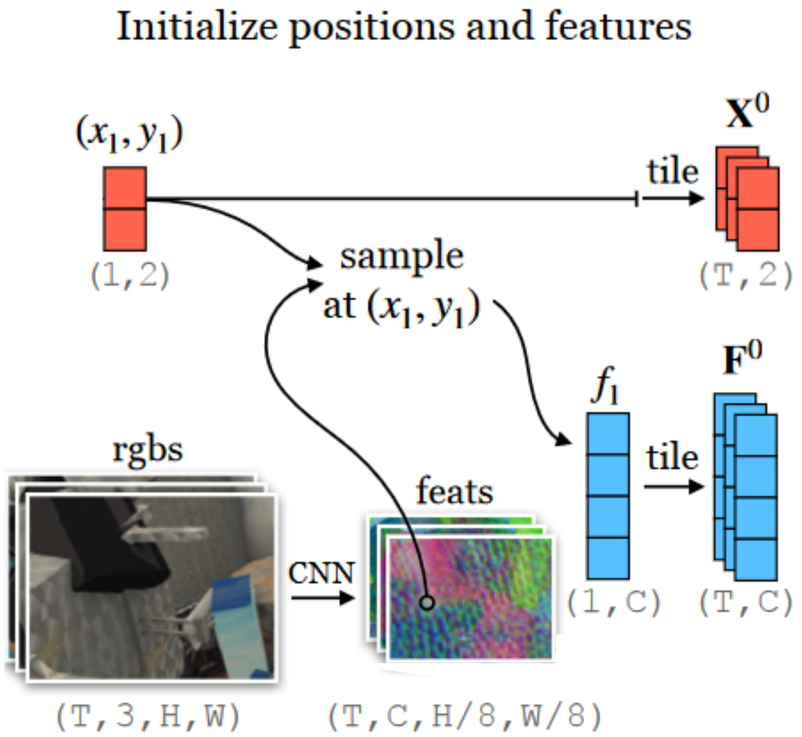


(T,3,H,W)

Input video

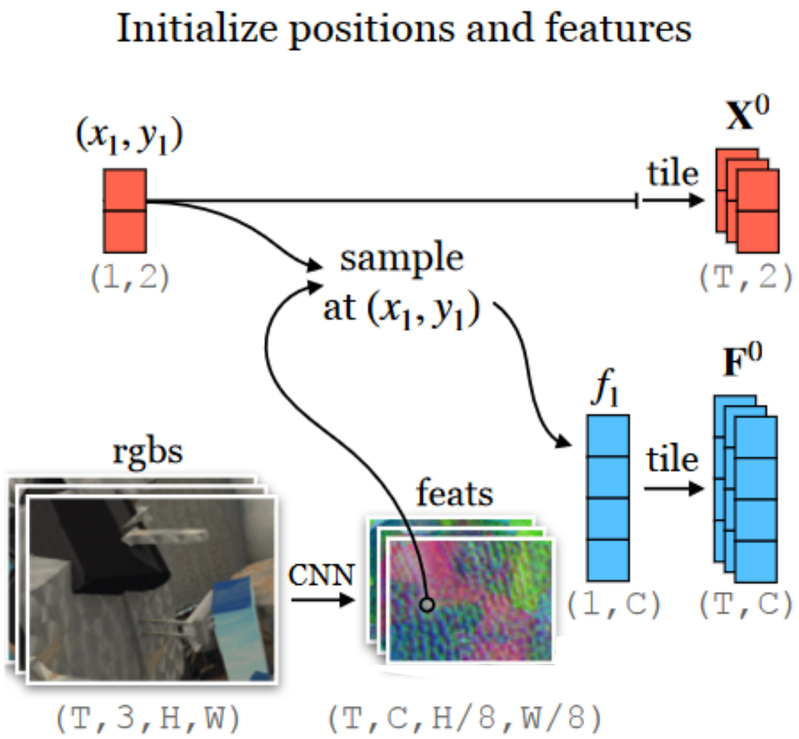
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



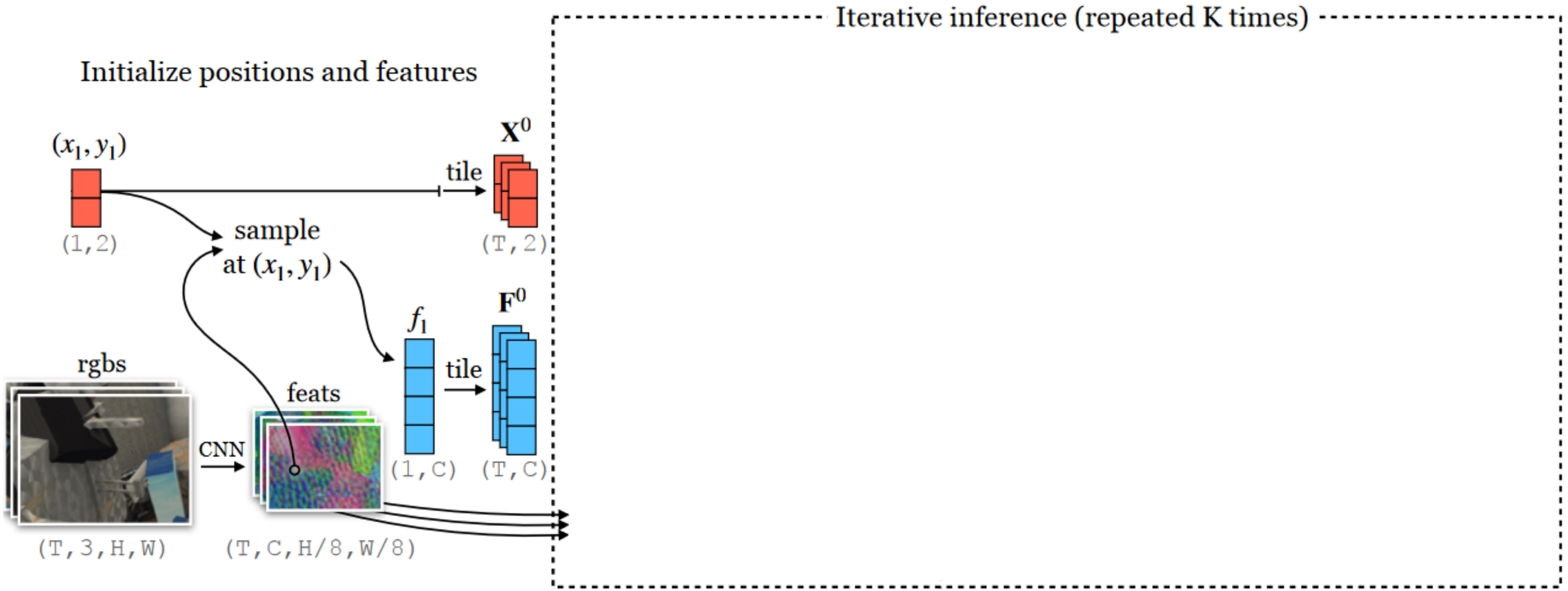
Initialize constant trajectory

Initialize constant appearance

Learned features

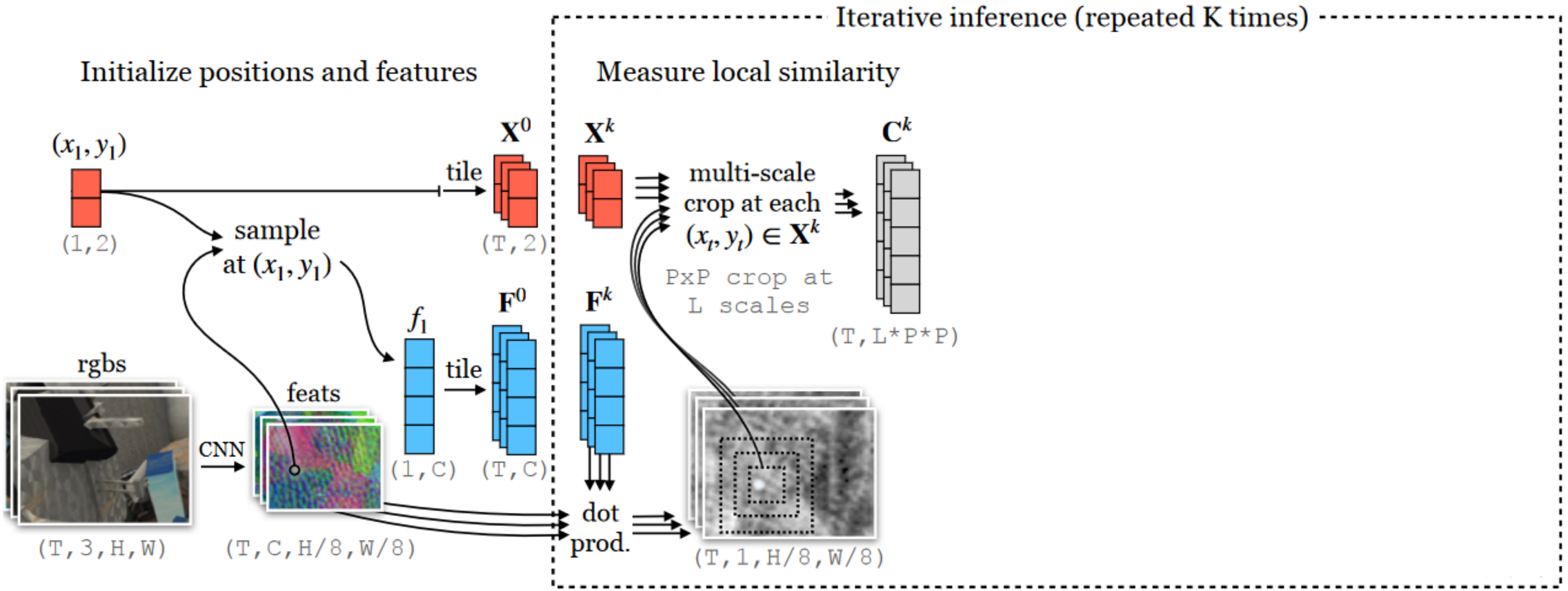
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



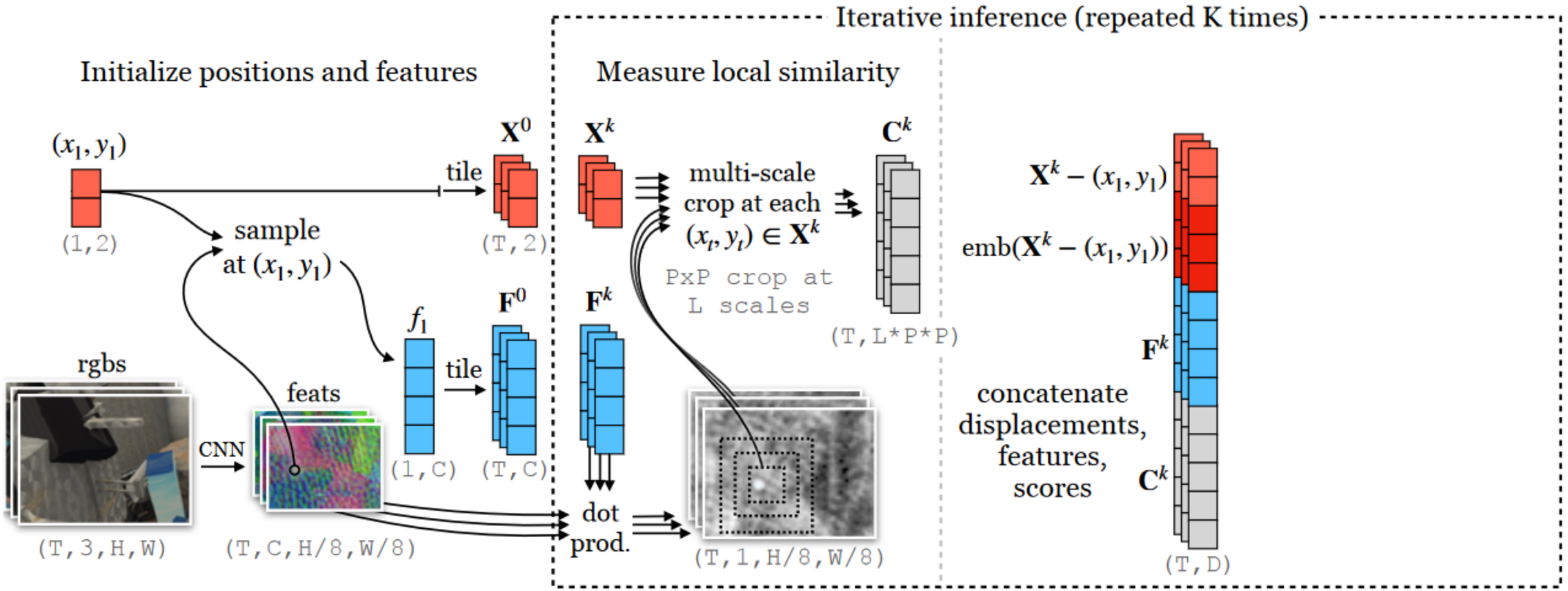
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



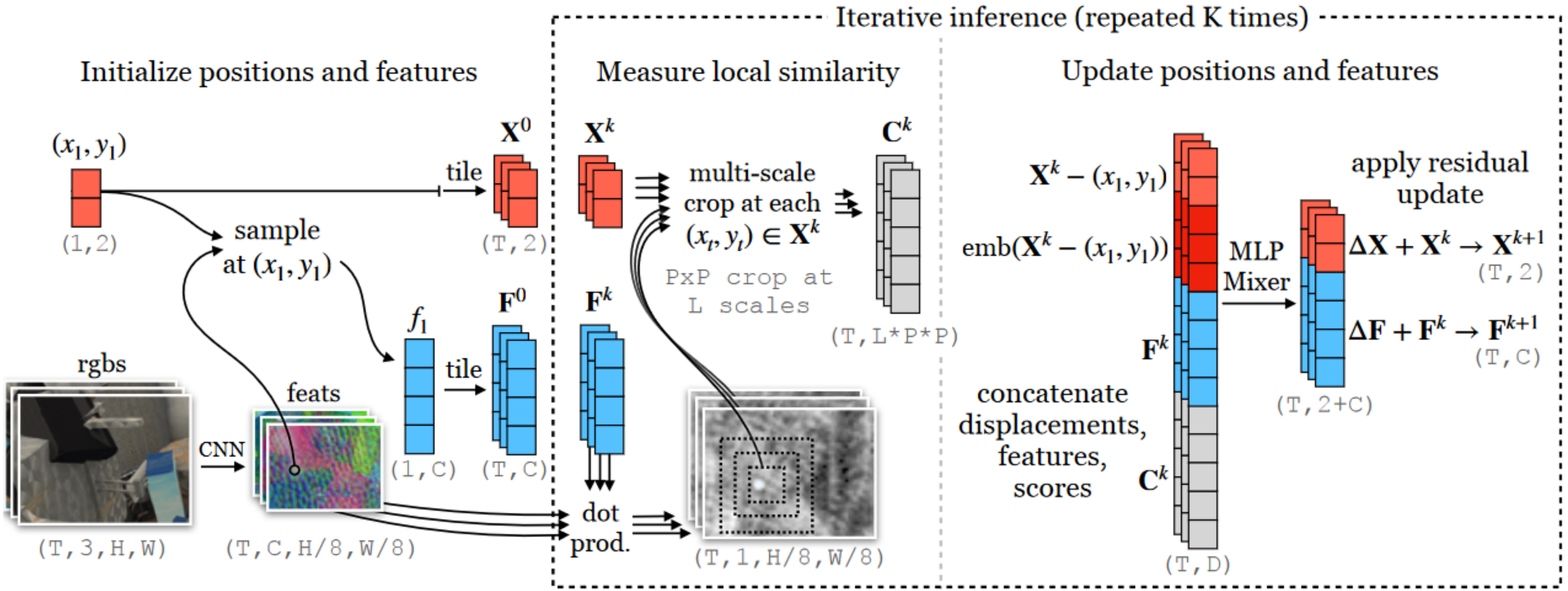
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



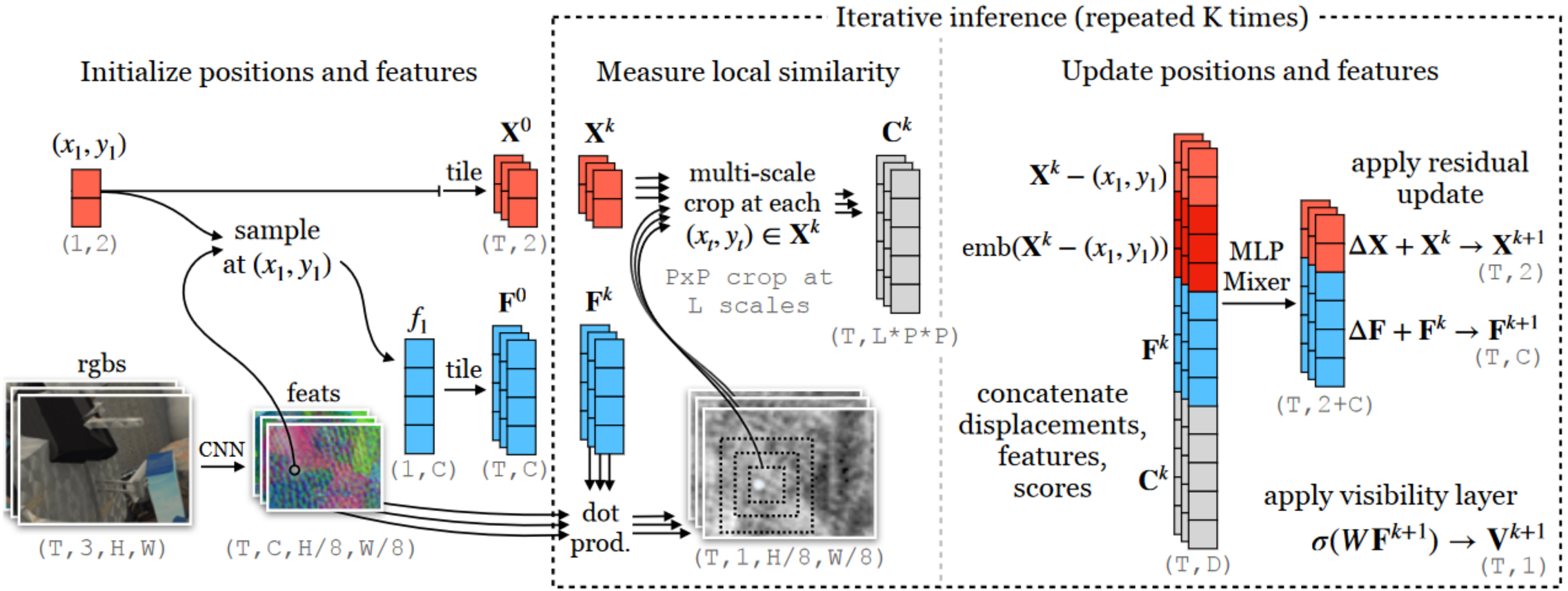
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

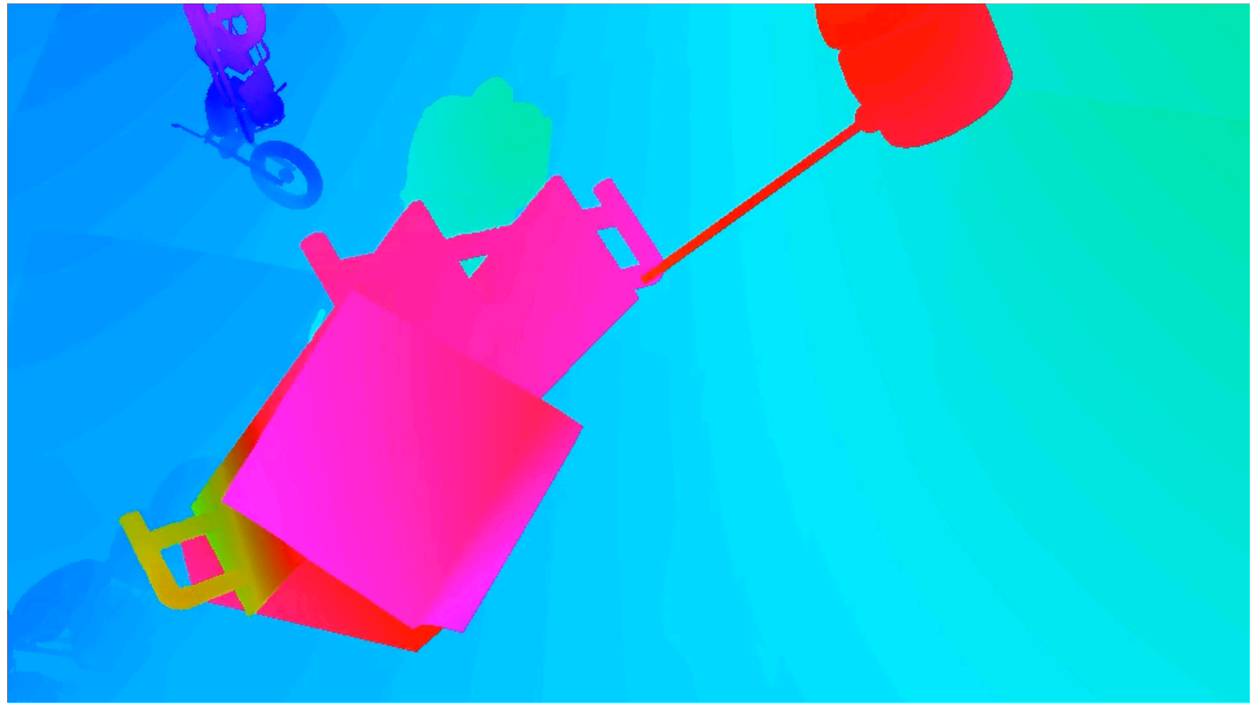
Particle Video Revisited



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

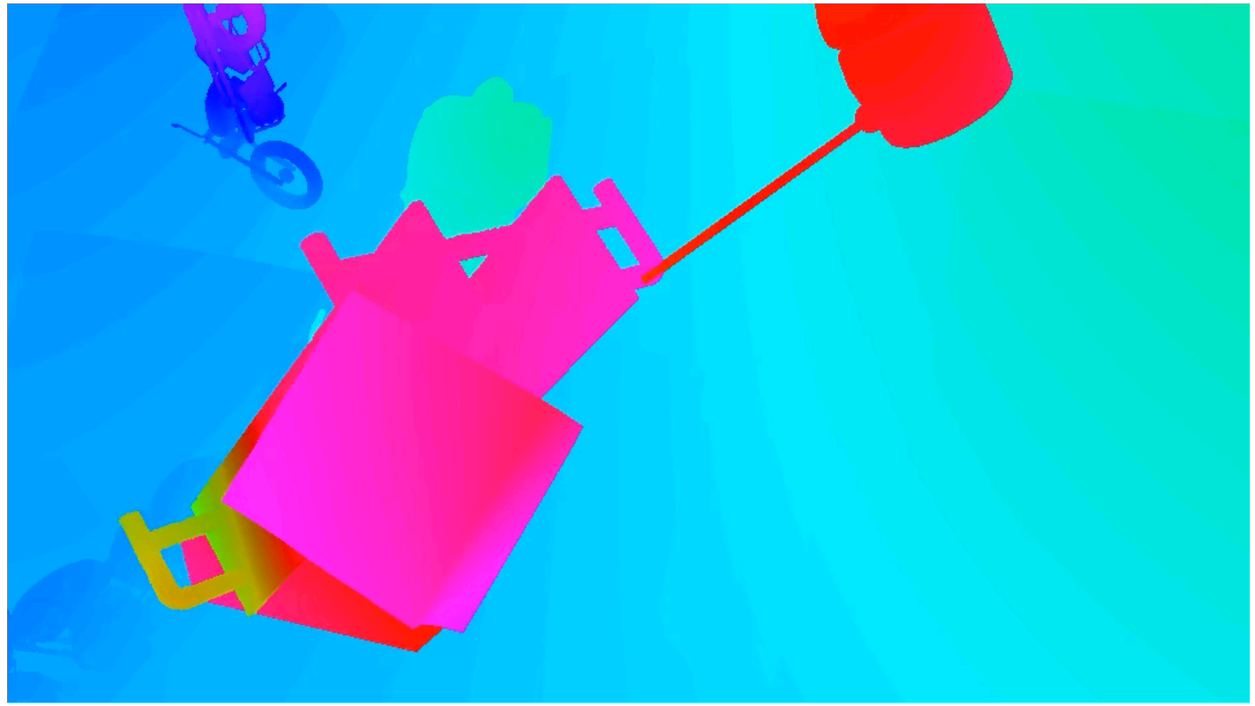
Trained Fully Supervised on Synthetic Data



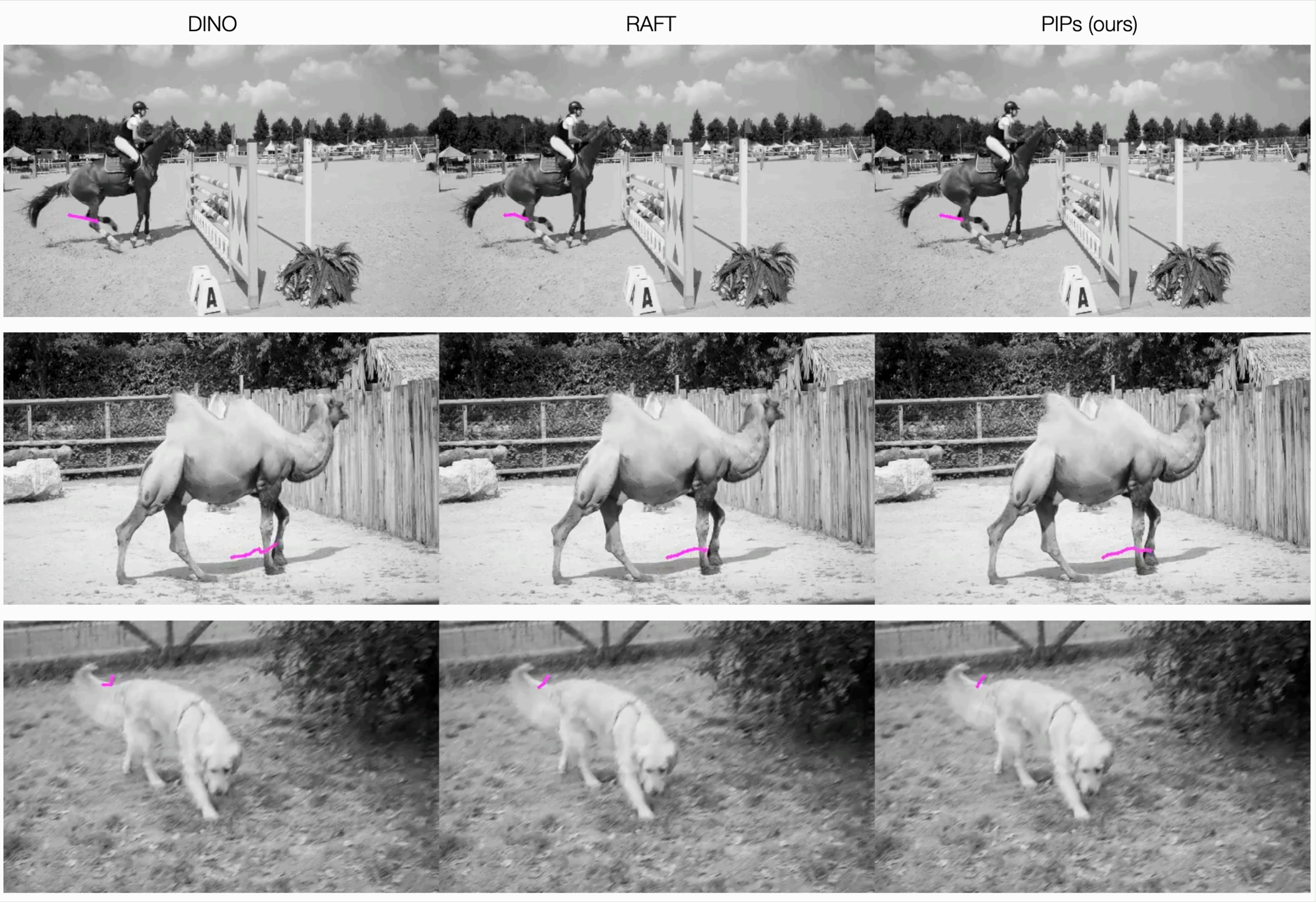


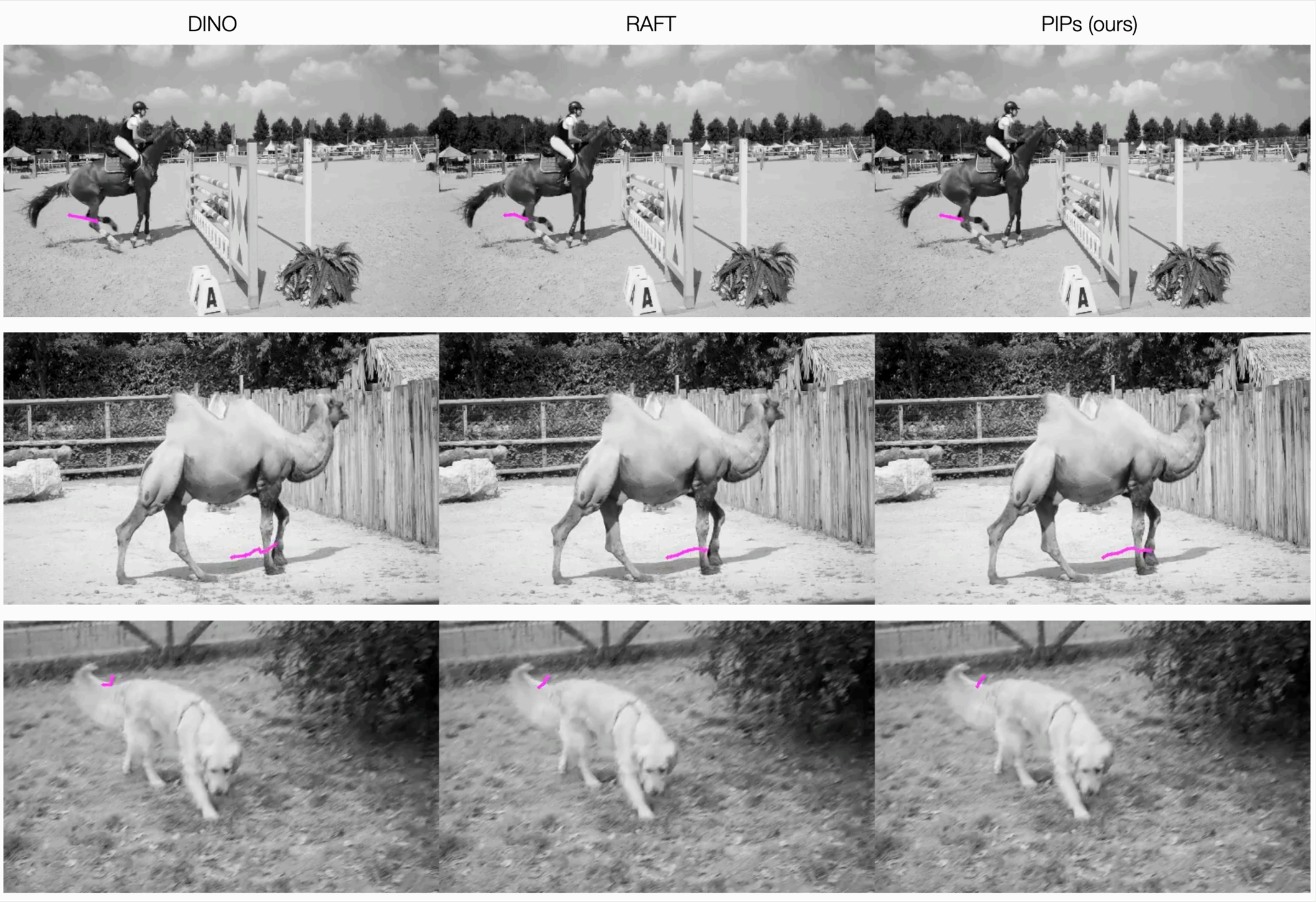
Trained Fully Supervised on Synthetic Data





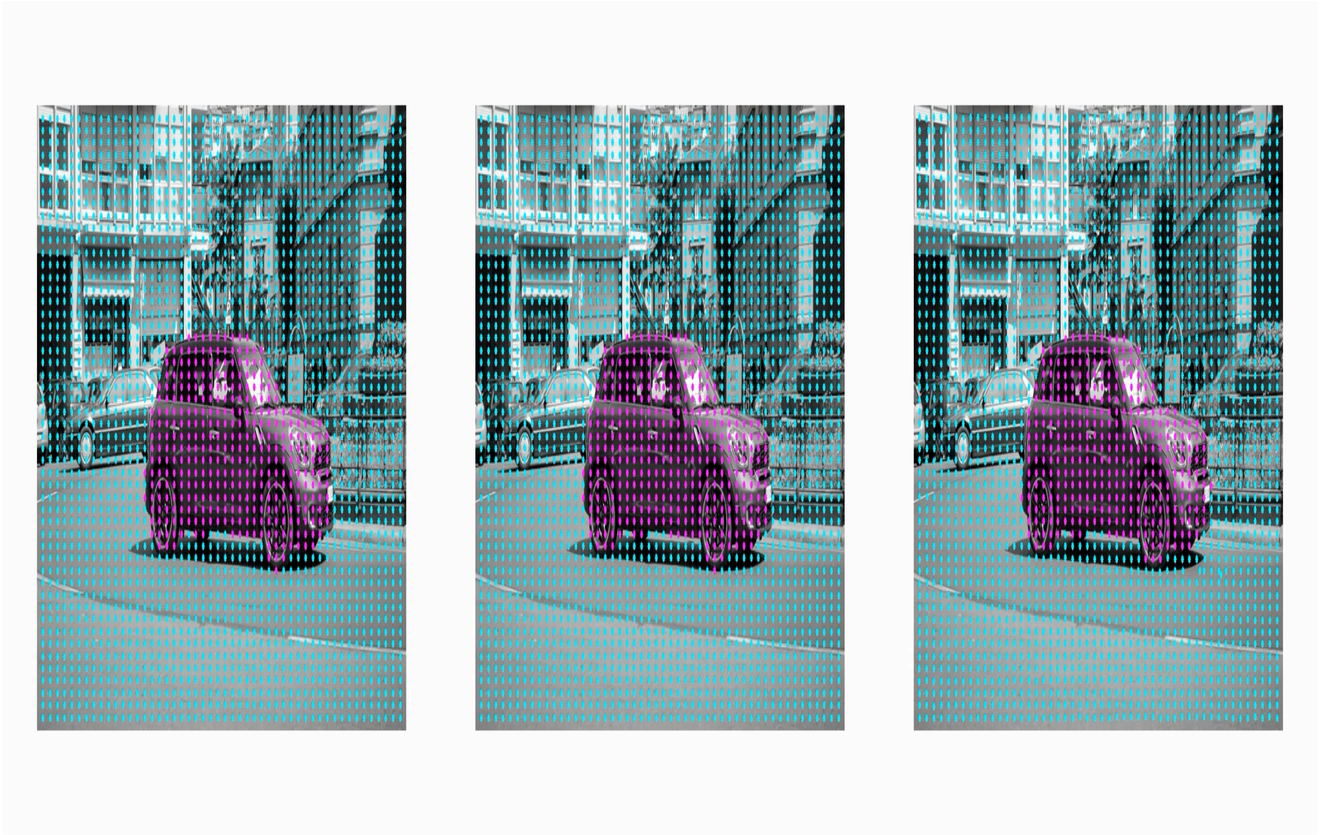
A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation, Mayer et al., 2016


DINO
RAFT
PIPs (ours)


DINO
RAFT
PIPs (ours)

From a different paper:

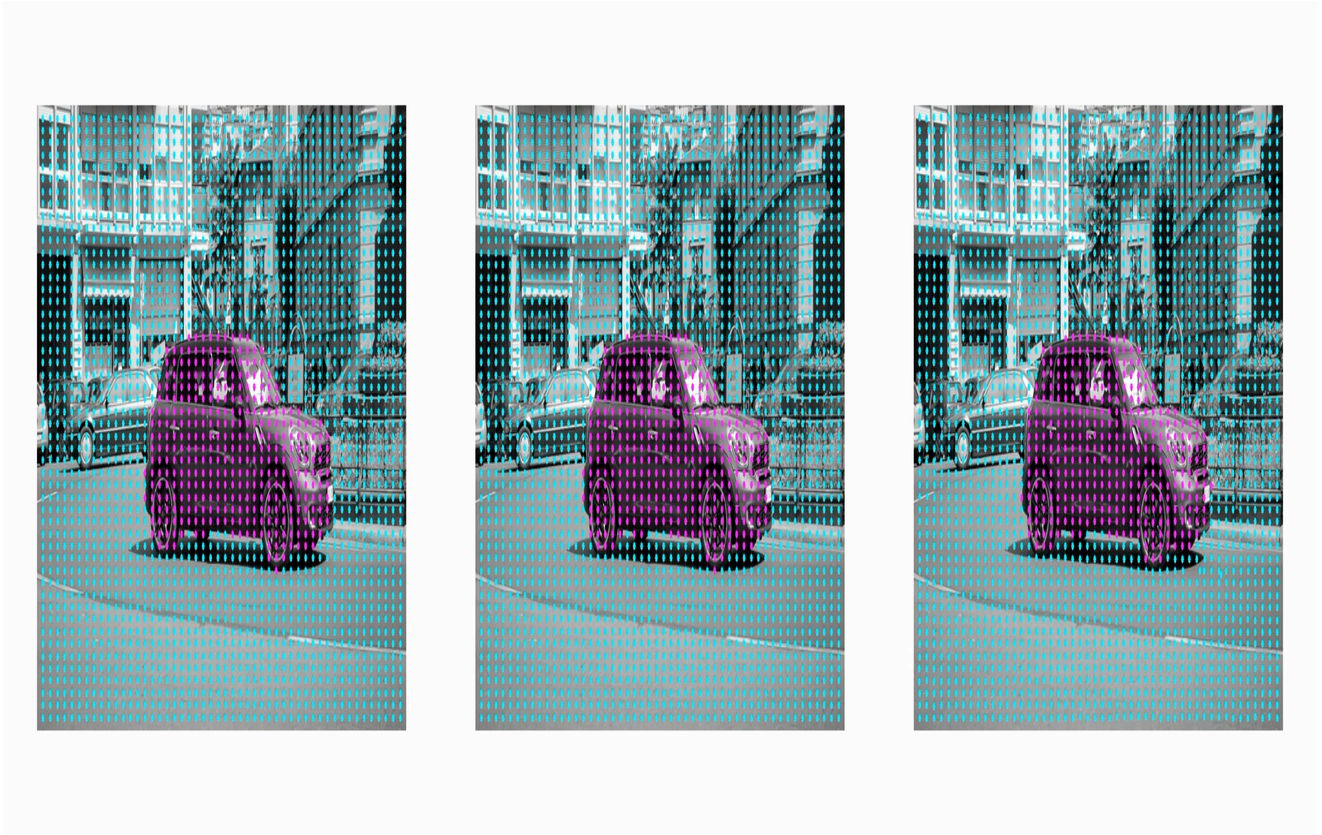
Limitation: Single Point Tracking



- Background points
- Object points

From a different paper:

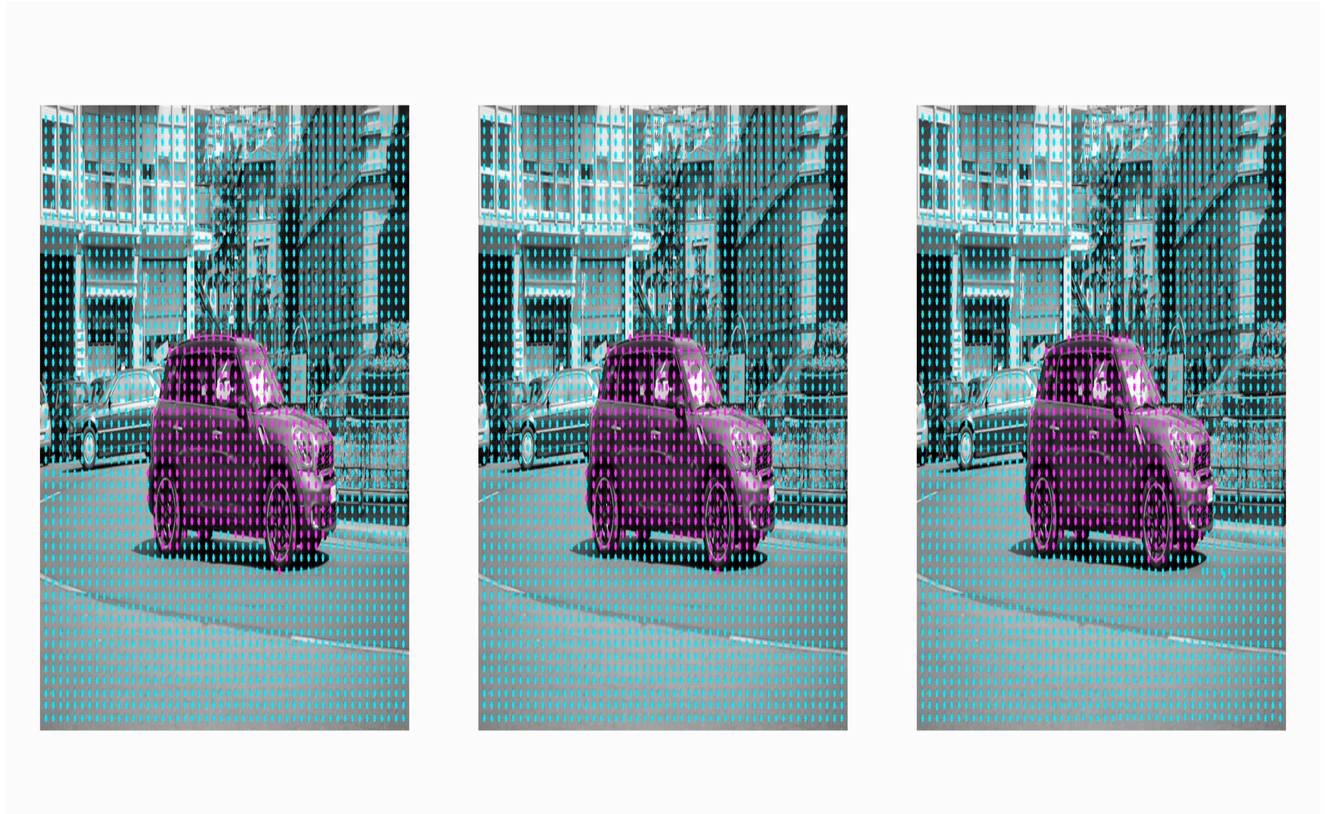
Limitation: Single Point Tracking



- Background points
- Object points

From a different paper:

Limitation: Single Point Tracking



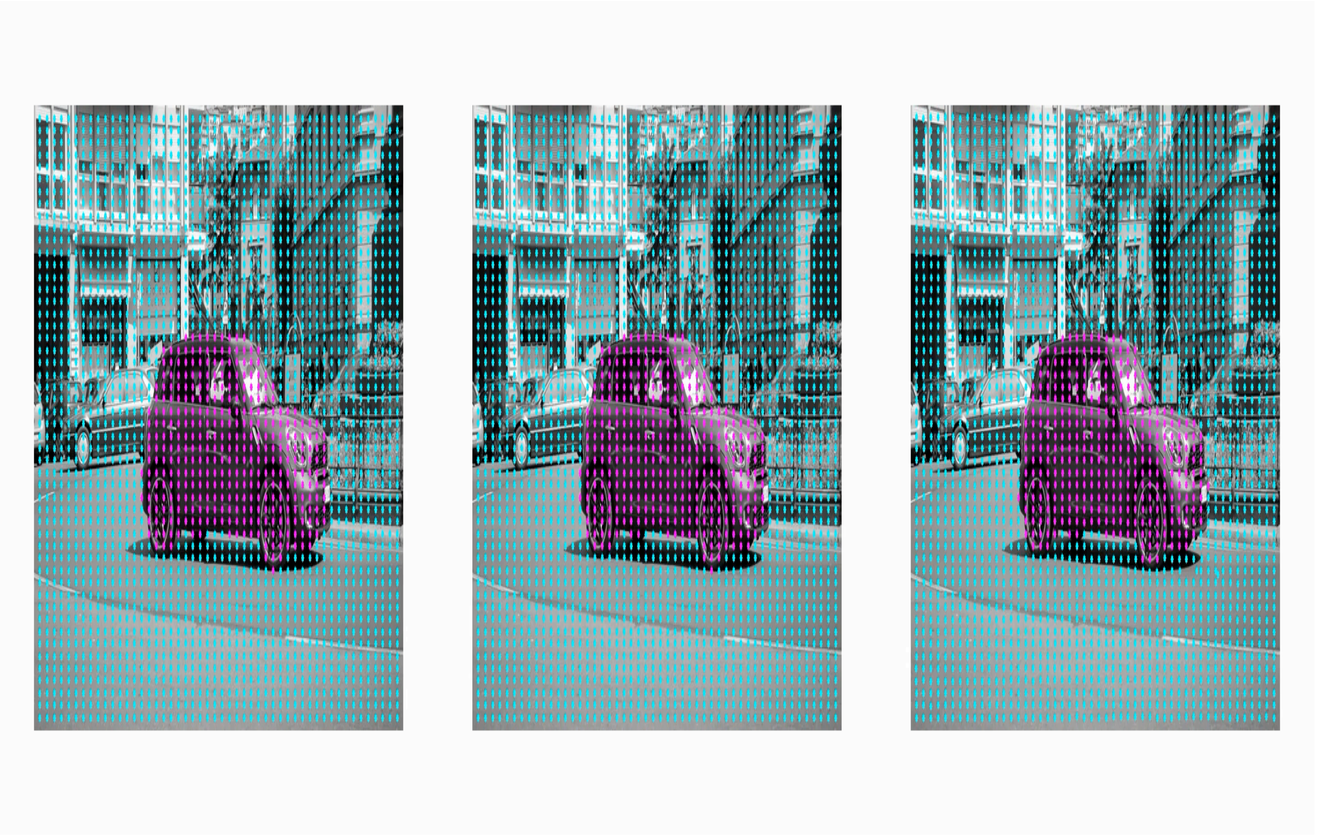
Background points

Object points

Tracking single points lacks global consistency

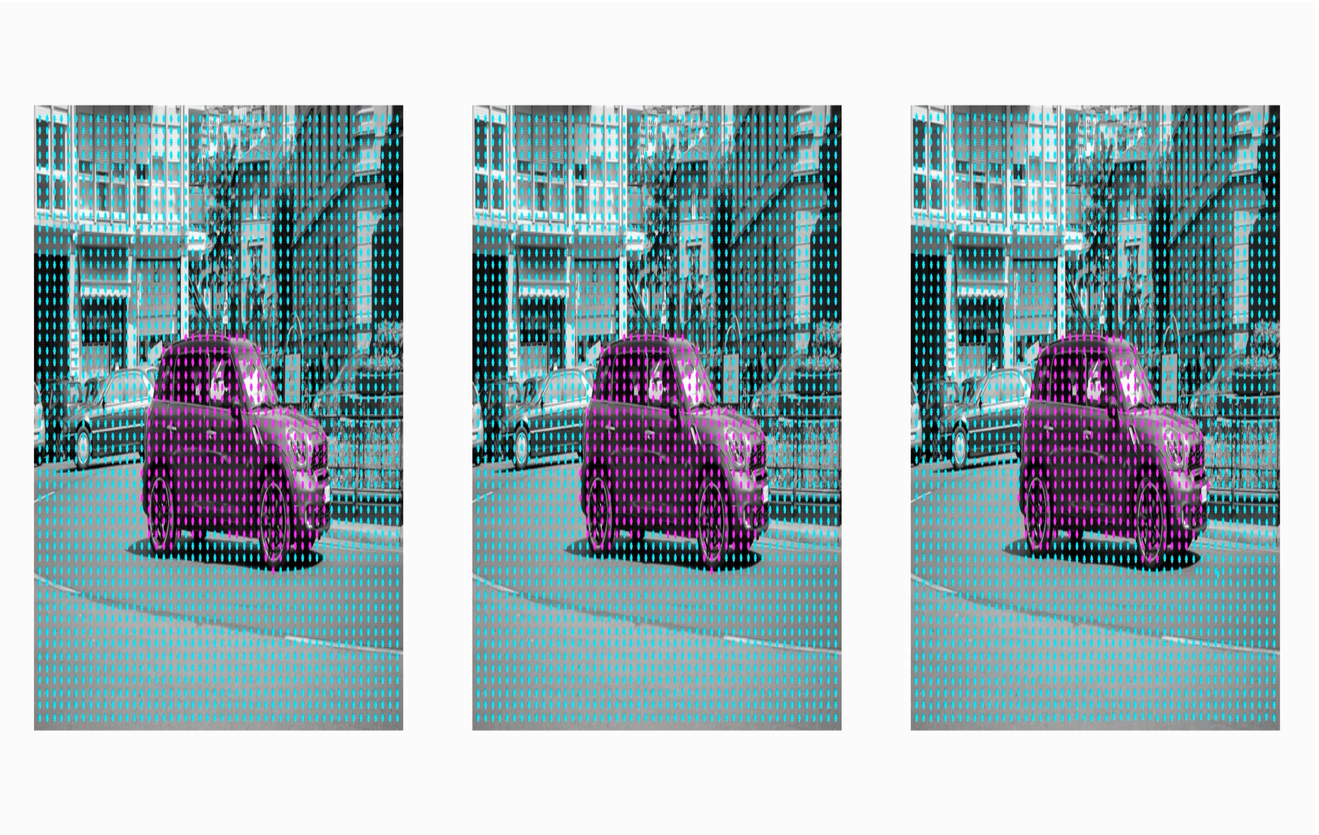
Points drift relative to each other

Tracking with Optical Flow



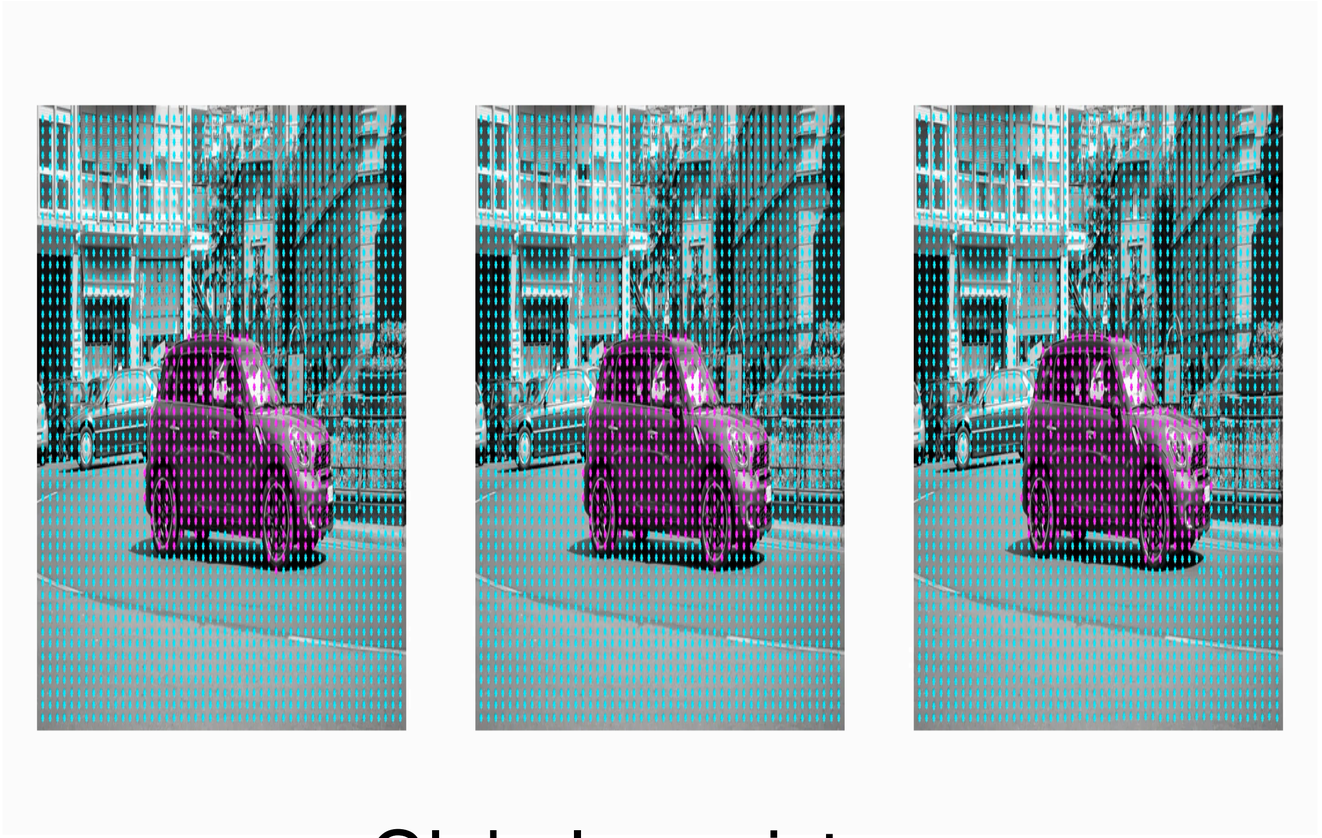
- Background points
- Object points

Tracking with Optical Flow



- Background points
- Object points

Tracking with Optical Flow



- Background points
- Object points

Global consistency

Occlusion problem: background accumulates on the object

CoTracker: It is Better to Track Together

Nikita Karaev[1,2], Ignacio Rocco[1], Benjamin Graham[1], Natalia Neverova[1], Andrea Vedaldi[1], and Christian Rupprecht[2]

[1] Meta AI
[2] Visual Geometry Group, University of Oxford
https://co-tracker.github.io/
nikita@robots.ox.ac.uk

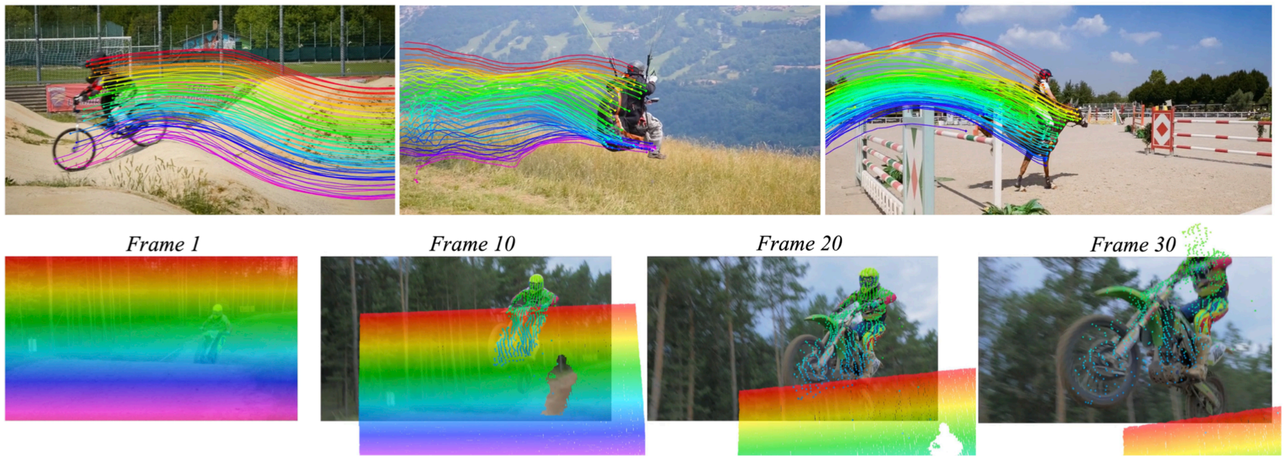


Fig. 1: CoTracker is a new quasi-dense point tracker. It can track 70k points jointly on a single GPU, exploiting dependencies between different tracks to enhance performance. It has exceptional long-term tracking performance even in the presence of occlusions or when points leave the camera view (examples from DAVIS [46], top row displays object tracks, bottom row shows displacements on a dense regular grid. In frame 30, CoTracker successfully tracks the driver's head even when it exists the camera view).

https://co-tracker.github.io/





CoTracker: It is Better to Track Together

Nikita Karaev[1,2], Ignacio Rocco[1], Benjamin Graham[1], Natalia Neverova[1], Andrea Vedaldi[1], and Christian Rupprecht[2]

[1] Meta AI
[2] Visual Geometry Group, University of Oxford
https://co-tracker.github.io/
nikita@robots.ox.ac.uk

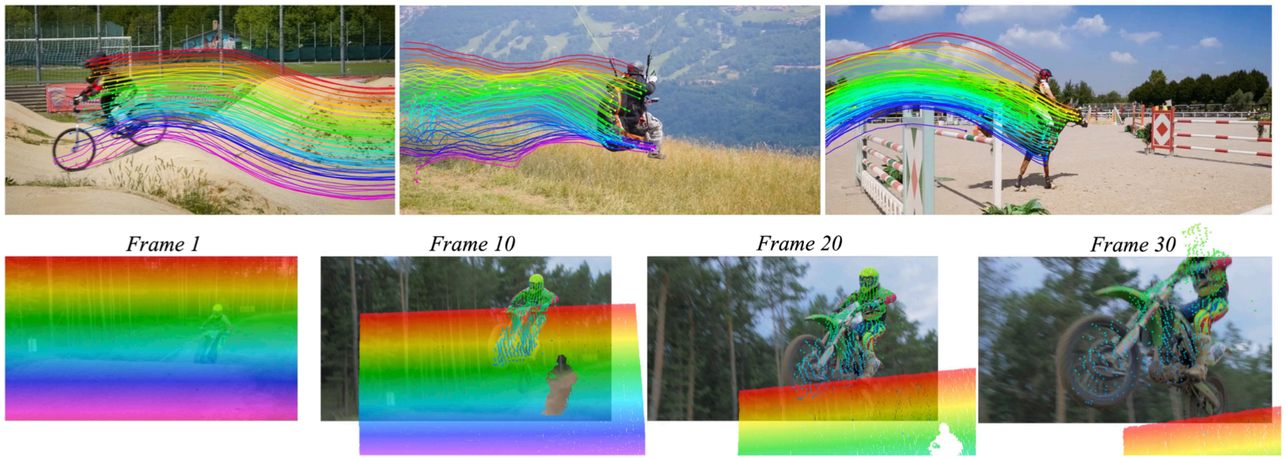


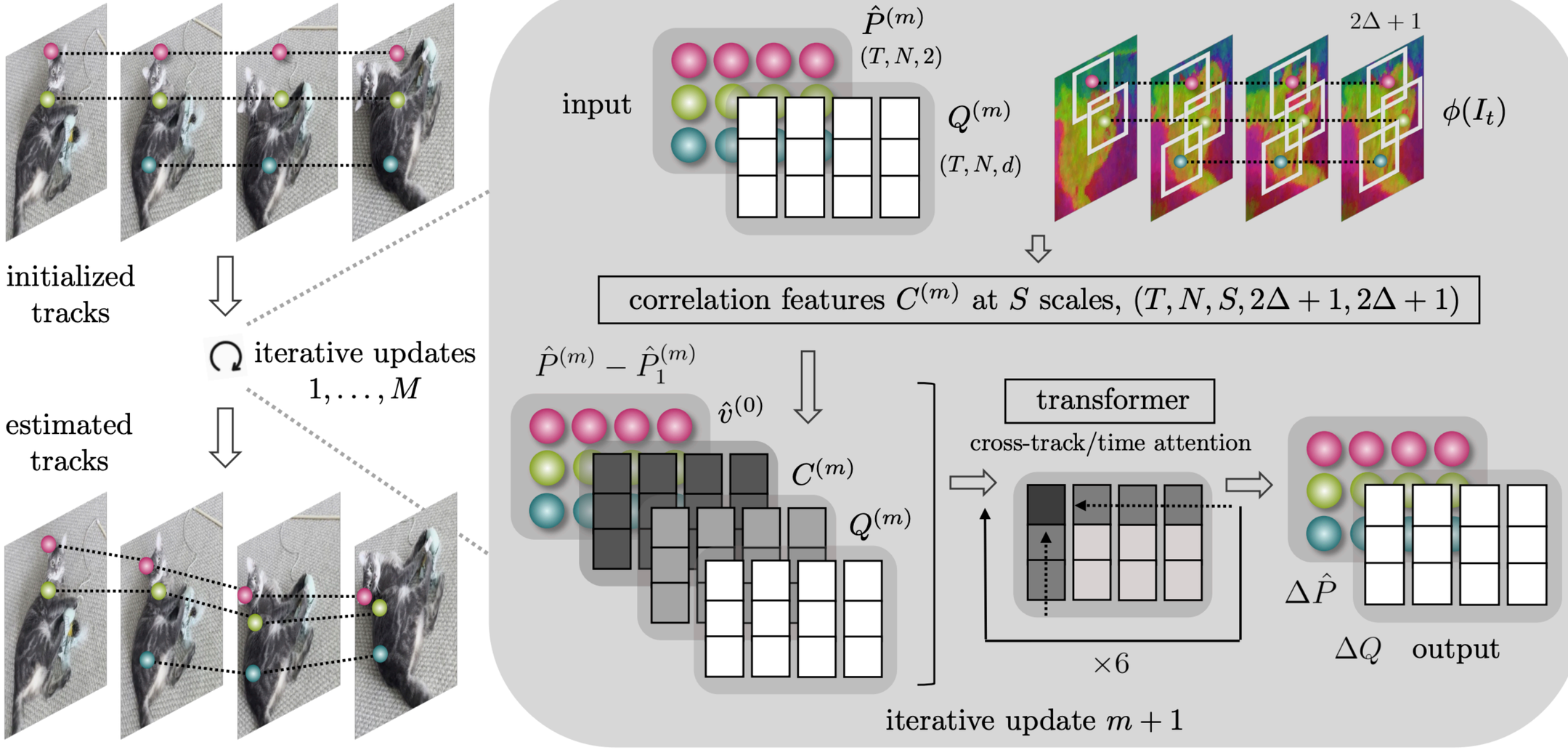
Fig. 1: CoTracker is a new quasi-dense point tracker. It can track 70k points jointly on a single GPU, exploiting dependencies between different tracks to enhance performance. It has exceptional long-term tracking performance even in the presence of occlusions or when points leave the camera view (examples from DAVIS [46], top row displays object tracks, bottom row shows displacements on a dense regular grid. In frame 30, CoTracker successfully tracks the driver's head even when it exists the camera view).

https://co-tracker.github.io/





CoTracker: It is Better to Track Together



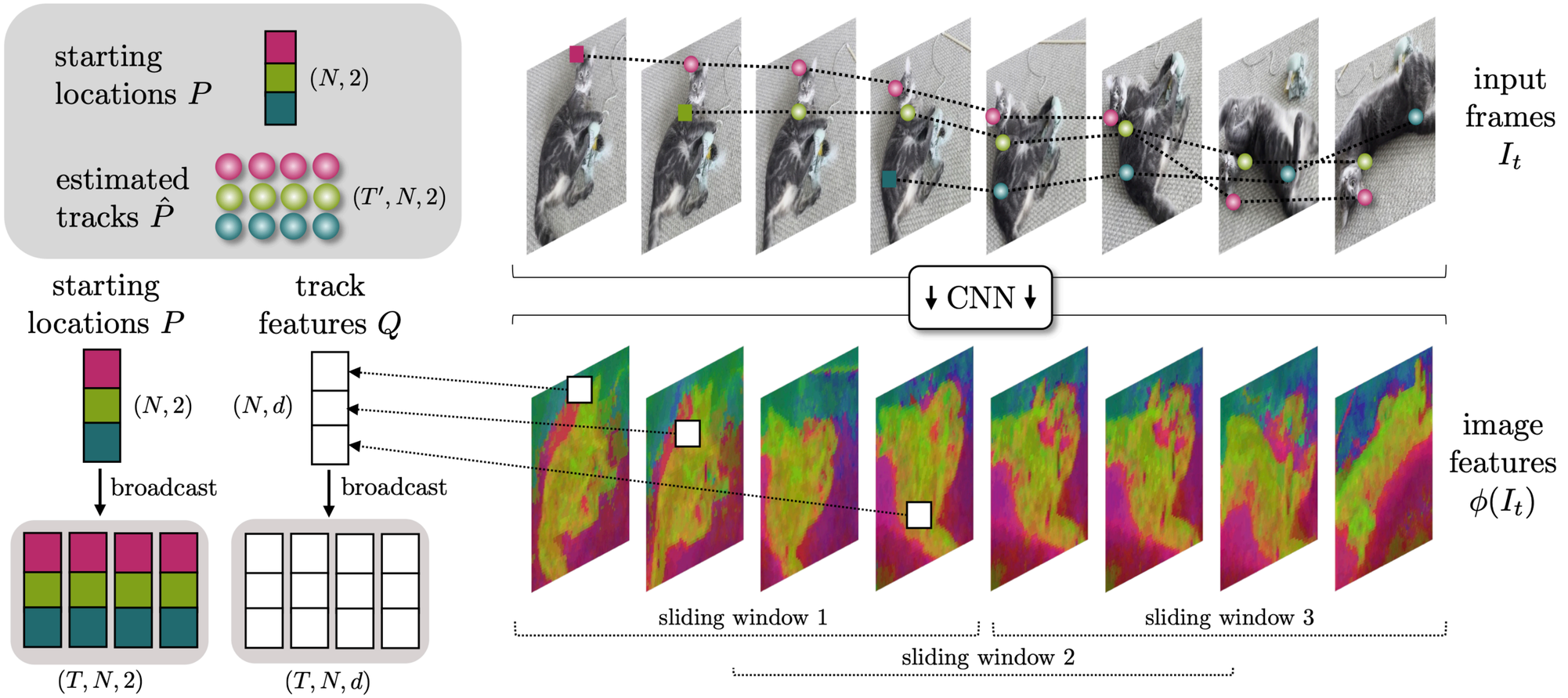


Fig. 3: CoTracker architecture. We compute convolutional features $\phi(I_t)$ for every frame and process them with sliding windows. To initialize track features Q, we bilinearly sample from $\phi(I_t)$ with starting point locations P. Locations P also serve to initialize estimated tracks $\hat{P}$. See Fig. 4 for a visualization of one sliding window.

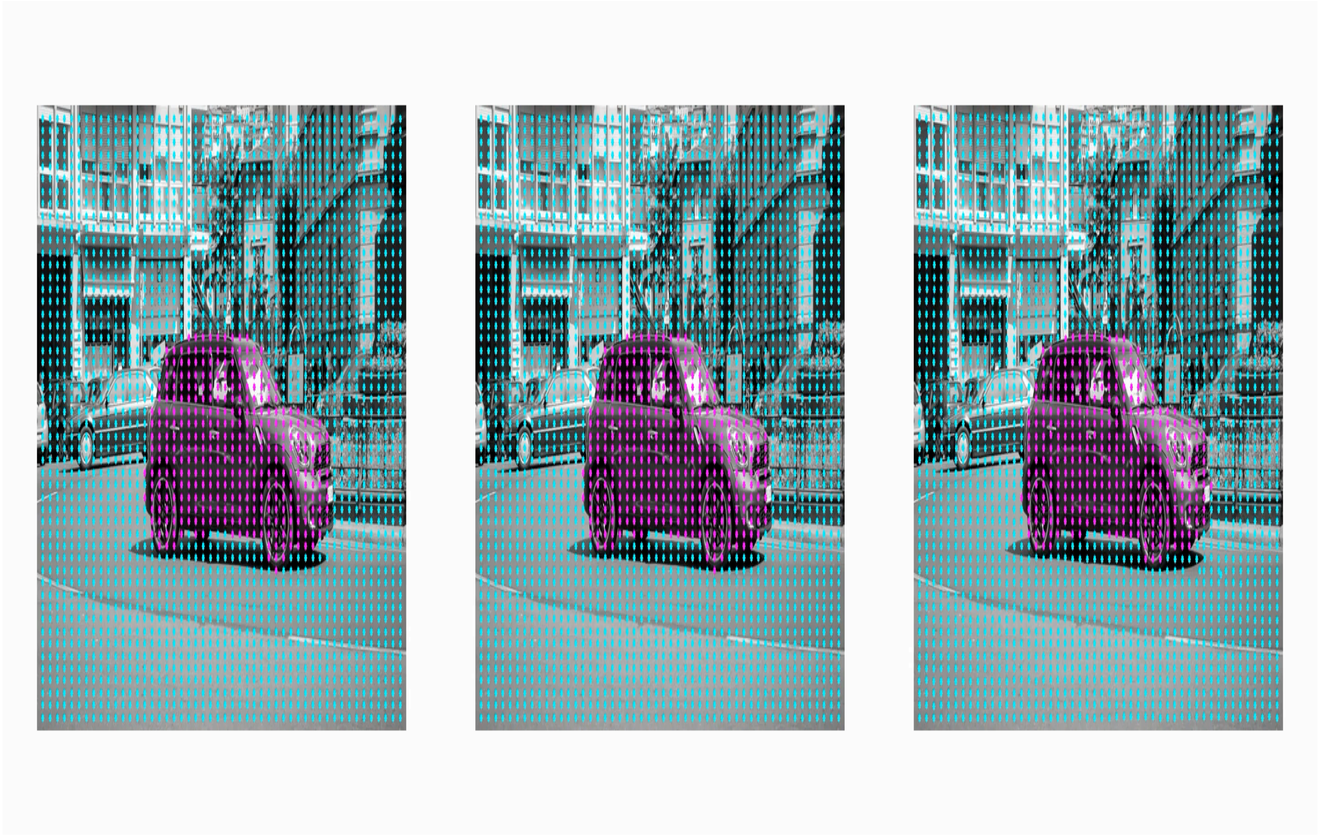
Tracking Points Together

- Global consistency from tracking many points
- Occlusion handling from long-term tracking
- Time-dependent appearance
- Train on synthetic data only

- Main mechanism:
  - Mid-range sliding window
  - Iterated updates
  - 4D cost volume
  - Transformer architecture
    - Input Tokens: position, visibility, appearance and correlation of the tracks.(with positional encoding)
    - Output: updates to location, appearance
  - Factorized attention: space, time, and group (across tracks)
  - ~registers (proxy tokens)

CoTracker: It is Better to Track Together, N Karaev et al., 2023

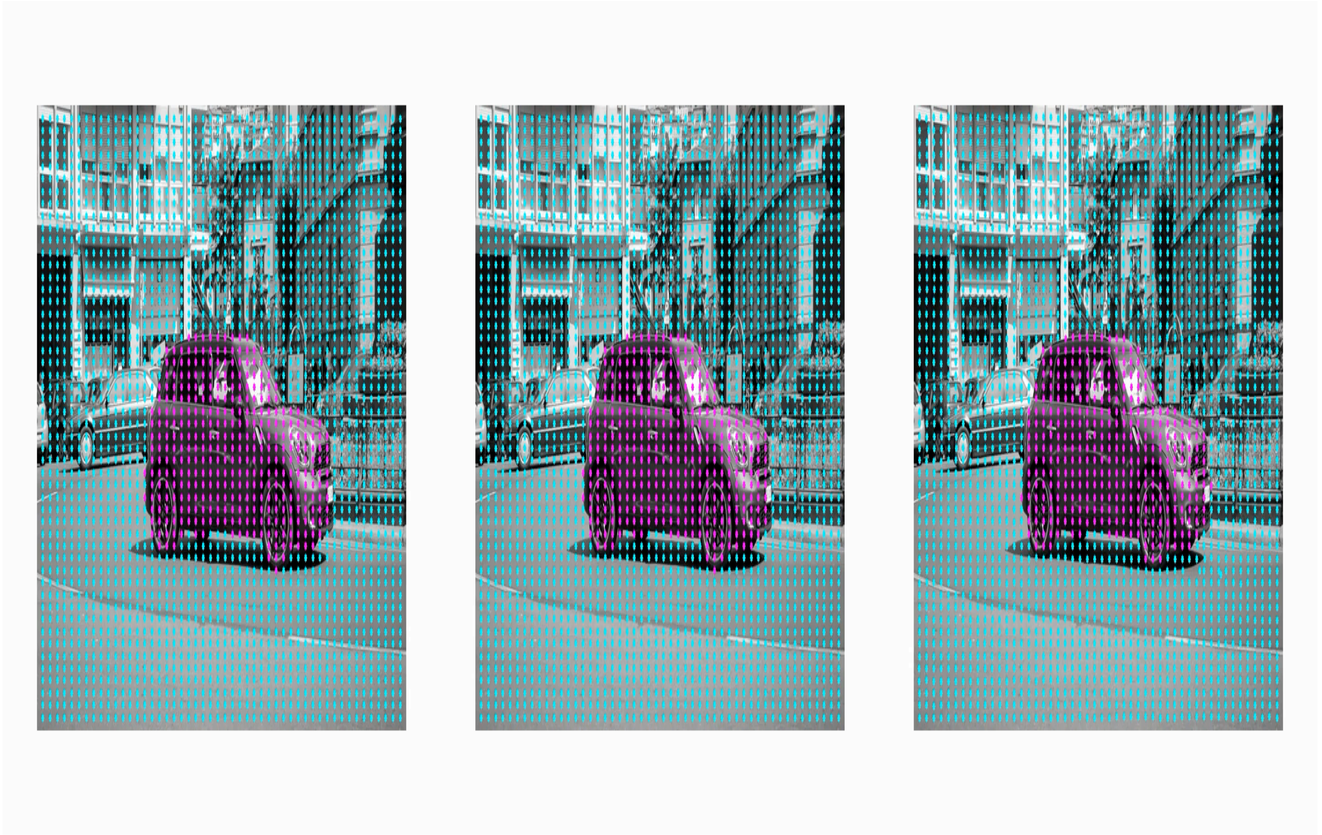
Tracking Points Together



- Background points
- Object points

- Better global consistency
- Better occlusion handling

Tracking Points Together



- Background points
- Object points

- Better global consistency
- Better occlusion handling

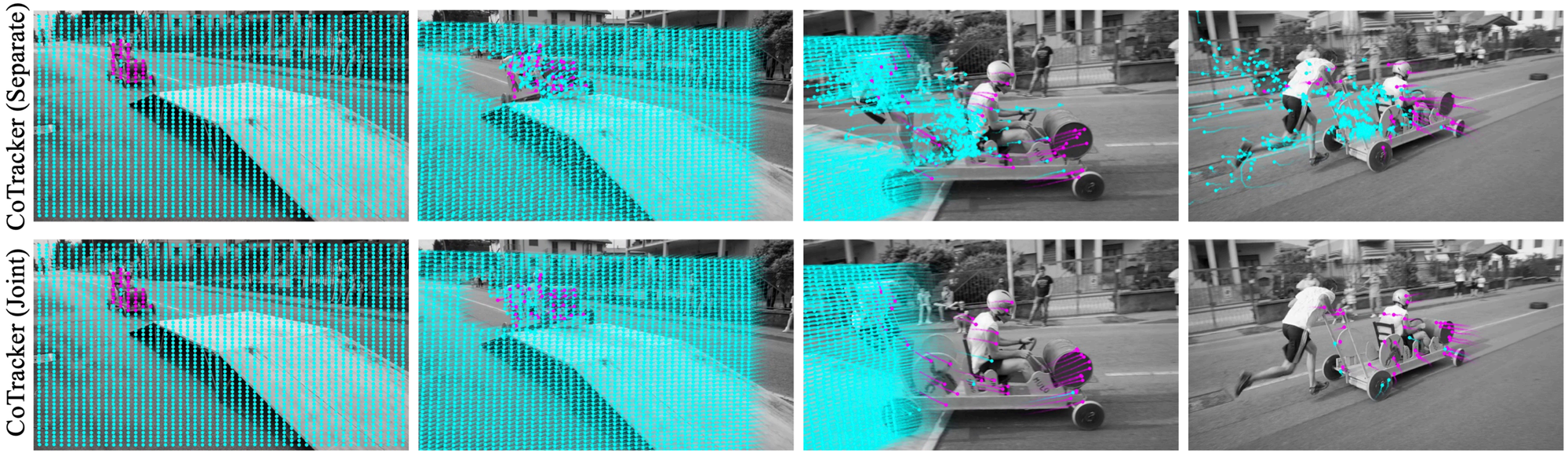


Fig. 2: Non-joint tracking (top row) vs. joint tracking (bottom). Background points are colored cyan while foreground points are magenta. Non-joint point tracking causes background points to follow object motion (cyan points in later frames), while tracking points together results in better tracks for foreground and background points.

Tracking Points Together



Tracking Points Together



Tracking Points Together



Tracking Points Together



Registers

- Consider Deep learning dense features
- Sometimes Neural Networks need to store global information
- They may repurpose some pixels to store global information
  - Kind of like steganography
- Leads to spiky artifacts (orders of magnitude higher than other pixels)
- Solution: extra tokens

VISION TRANSFORMERS NEED REGISTERS

Timothée Darcet[1,2] **Maxime Oquab**[1] **Julien Mairal**[2] **Piotr Bojanowski**[1]

[1] FAIR, Meta
[2] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France

ABSTRACT

Transformers have recently emerged as a powerful tool for learning visual representations. In this paper, we identify and characterize artifacts in feature maps of both supervised and self-supervised ViT networks. The artifacts correspond to high-norm tokens appearing during inference primarily in low-informative background areas of images, that are repurposed for internal computations. We propose a simple yet effective solution based on providing additional tokens to the input sequence of the Vision Transformer to fill that role. We show that this solution fixes that problem entirely for both supervised and self-supervised models, sets a new state of the art for self-supervised visual models on dense visual prediction tasks, enables object discovery methods with larger models, and most importantly leads to smoother feature maps and attention maps for downstream visual processing.

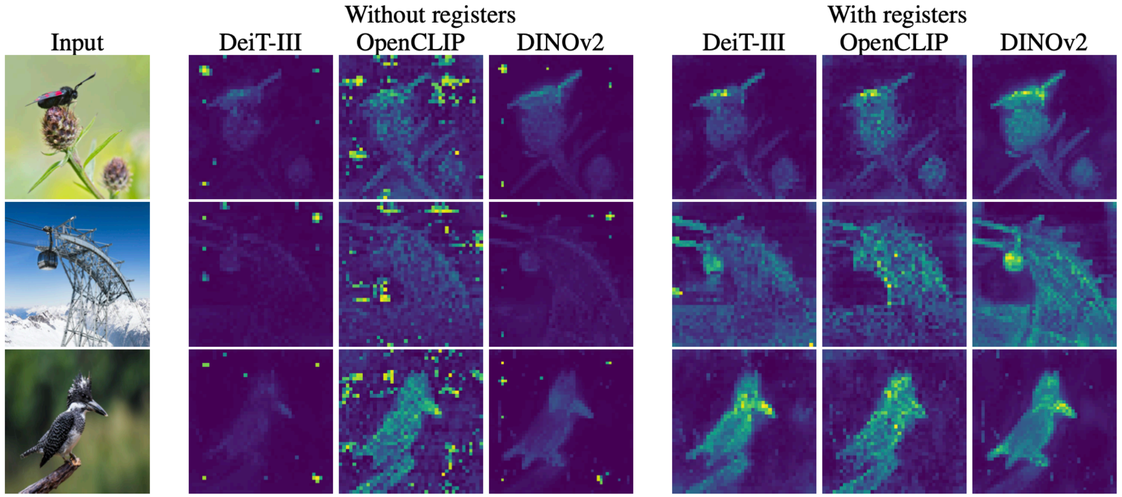


Figure 1: Register tokens enable interpretable attention maps in all vision transformers, similar to the original DINO method (Caron et al., 2021). Attention maps are calculated in high resolution for better visualisation. More qualitative results are available in appendix H.

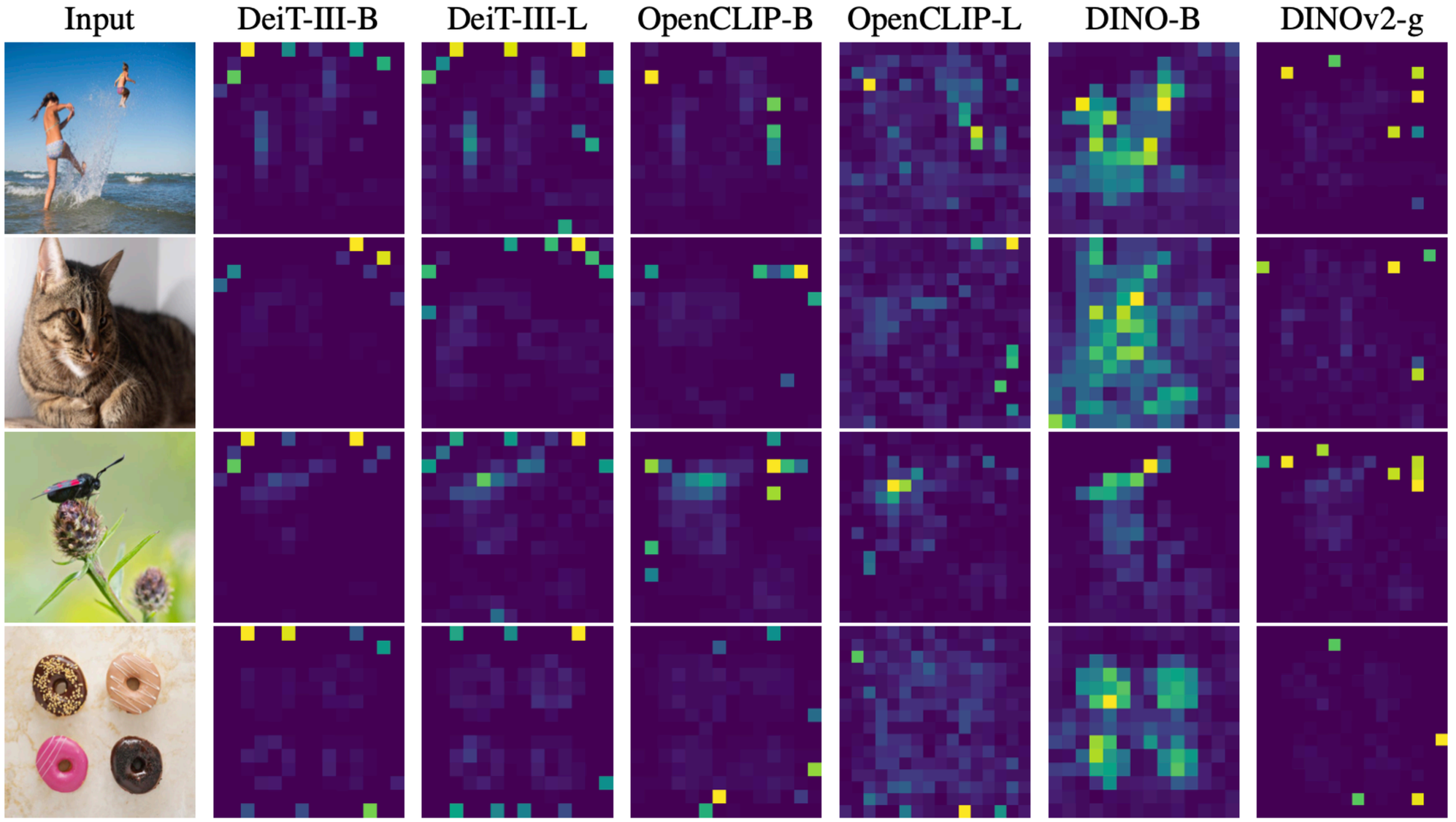


Figure 2: Illustration of artifacts observed in the attention maps of modern vision transformers. We consider ViTs trained with label supervision (DeiT-III), text-supervision (OpenCLIP) or self-supervision (DINO and DINOv2). Interestingly, all models but DINO exhibit peaky outlier values in the attention maps. The goal of this work is to understand and mitigate this phenomenon.

Counterpoint about trained registers

Vision Transformers Don't Need *Trained* Registers

Nick Jiang* **Amil Dravid*** **Alexei A. Efros** **Yossi Gandelsman**

UC Berkeley
*Equal contribution
{nickj,amil_dravid,aaefros,yossi_gandelsman}@berkeley.edu

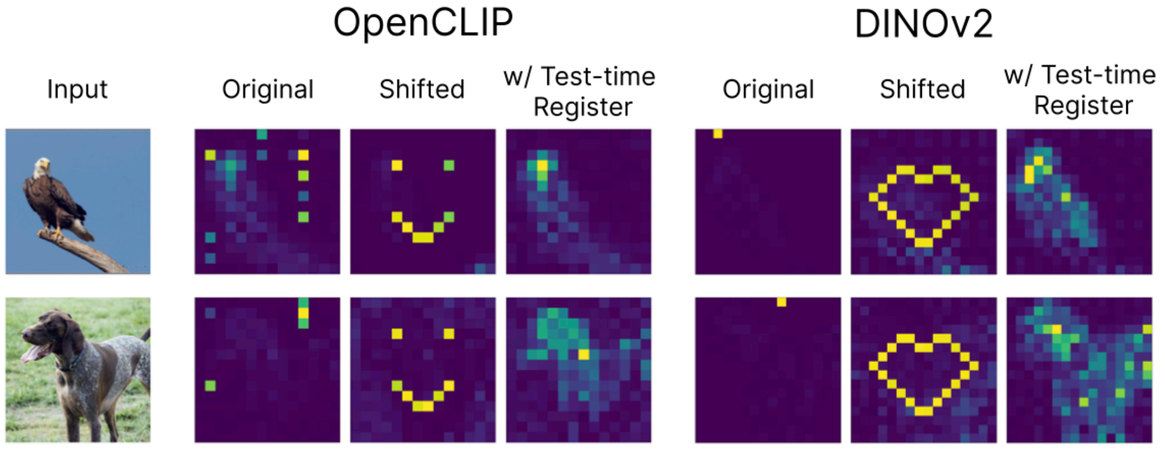


Figure 1: **Controlling high-norm tokens in Vision Transformers.** As shown in Darcet et al. (2024), high-norm outlier tokens emerge in ViTs and lead to noisy attention maps ("Original"). By identifying the mechanism responsible for their emergence, we demonstrate that we can shift these outlier tokens to arbitrary positions at test time ("Shifted"). Shifting the outlier tokens *outside* of the image mimics register behavior at test-time ("w/ Test-time Register"), resulting in more interpretable attention patterns and downstream performance comparable to models retrained with registers.

Abstract

We investigate the mechanism underlying a previously identified phenomenon in Vision Transformers – the emergence of high-norm tokens that lead to noisy attention maps (Darcet et al., 2024). We observe that in multiple models (e.g., CLIP, DINOv2), a sparse set of neurons is responsible for concentrating high-norm activations on outlier tokens, leading to irregular attention patterns and degrading downstream visual processing. While the existing solution for removing these outliers involves retraining models from scratch with additional learned *register tokens*, we use our findings to create a training-free approach to mitigate these artifacts. By shifting the high-norm activations from our discovered *register neurons* into an additional untrained token, we can mimic the effect of register tokens on a model already trained without registers. We demonstrate that our method produces cleaner attention and feature maps, enhances performance over base models across multiple downstream visual tasks, and achieves results comparable to models explicitly trained with register tokens. We then extend test-time registers to off-the-shelf vision-language models, yielding cleaner attention-based, text-to-image attribution. Finally, we outline a simple mathematical model that reflects the observed behavior of register neurons and high norm tokens. Our results suggest that test-time registers effectively take on the role of register tokens at test-time, offering a training-free solution for any pre-trained model released without them.[1]

https://avdravid.github.io/test-time-registers/

Families of correspondence problem(s)

	No 3D, motion	Static 3D
Dense	optical flow	Stereo
Sparse and smooth (+visibility)	Motion tracking	SLAM
Sparse and far	Panorama stitching	Structure From Motion

Flow Fields in 2D and 3D

Optical Flow





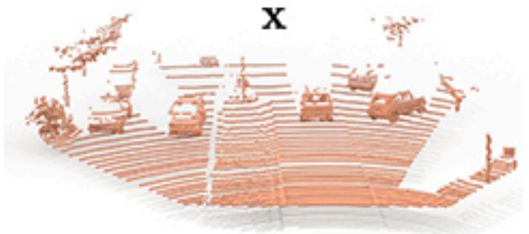


Optical Flow

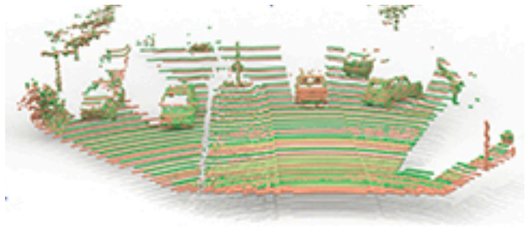
Given a pixel in frame 1, where does it go to in frame 2?

$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ A 2D flow field

Scene Flow





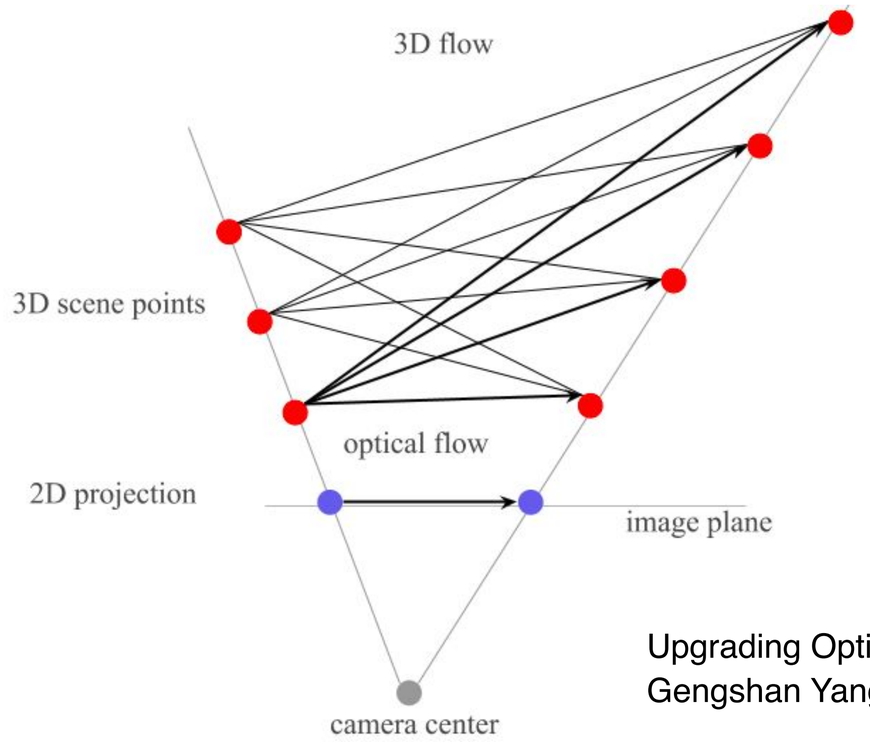


Per-point scene flow

Given a 3D point at time t1, where does it go at time t2?

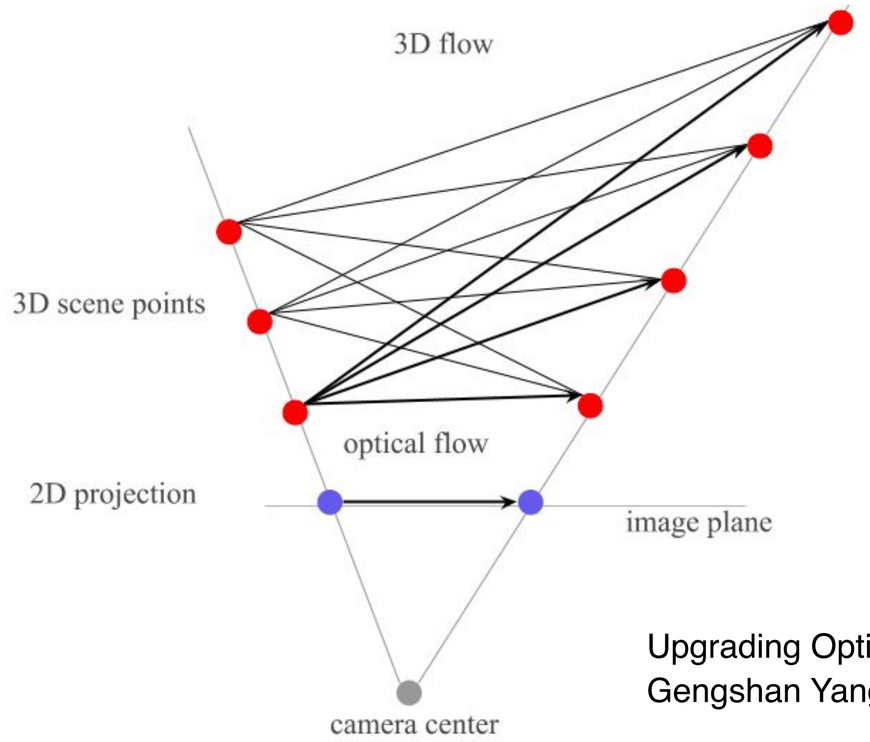
$\mathbb{R}^3 \rightarrow \mathbb{R}^3$ A 3D flow field

3D Scene Flow



Upgrading Optical Flow to 3D Scene Flow through Optical Expansion, Gengshan Yang et al, CVPR 2020

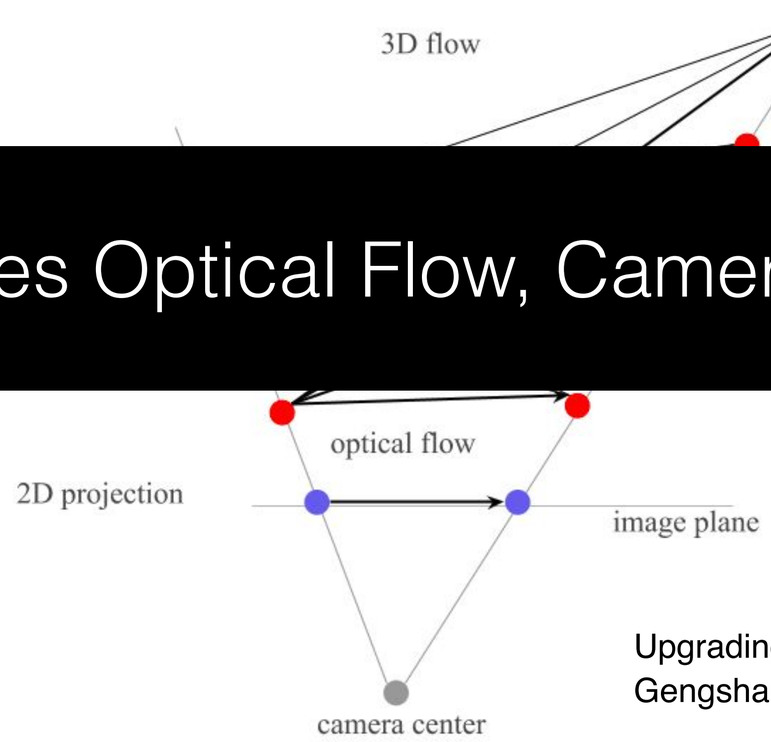
3D Scene Flow



Upgrading Optical Flow to 3D Scene Flow through Optical Expansion, Gengshan Yang et al, CVPR 2020

Observation: 2D flow constrains possible 3D flow (for visible points), and optical flow can be computed given 3D scene flow plus depth

3D Scene Flow

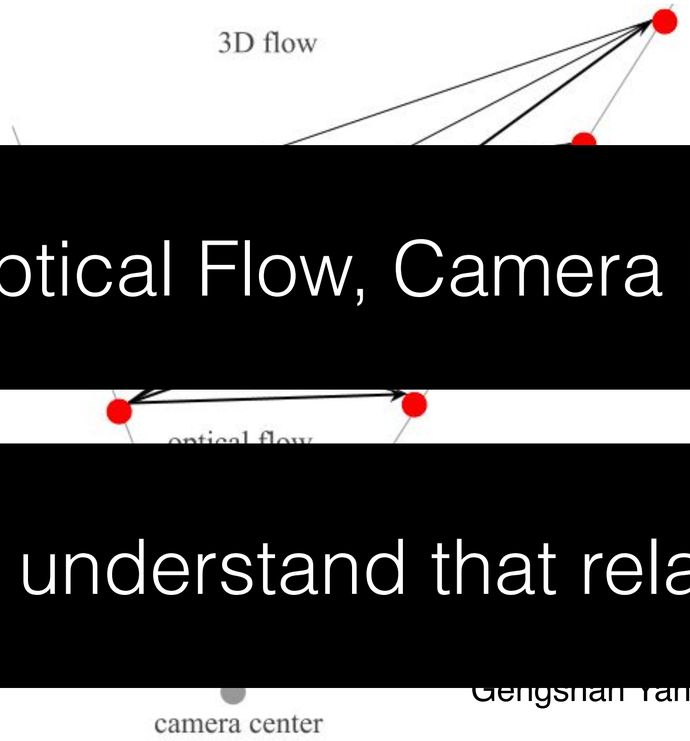


Scene Flow Relates Optical Flow, Camera Motion, and Depth!

Upgrading Optical Flow to 3D Scene Flow through Optical Expansion, Gengshan Yang et al, CVPR 2020

Observation: 2D flow constrains possible 3D flow (for visible points), and optical flow can be computed given 3D scene flow plus depth

3D Scene Flow



Scene Flow Relates Optical Flow, Camera Motion, and Depth!

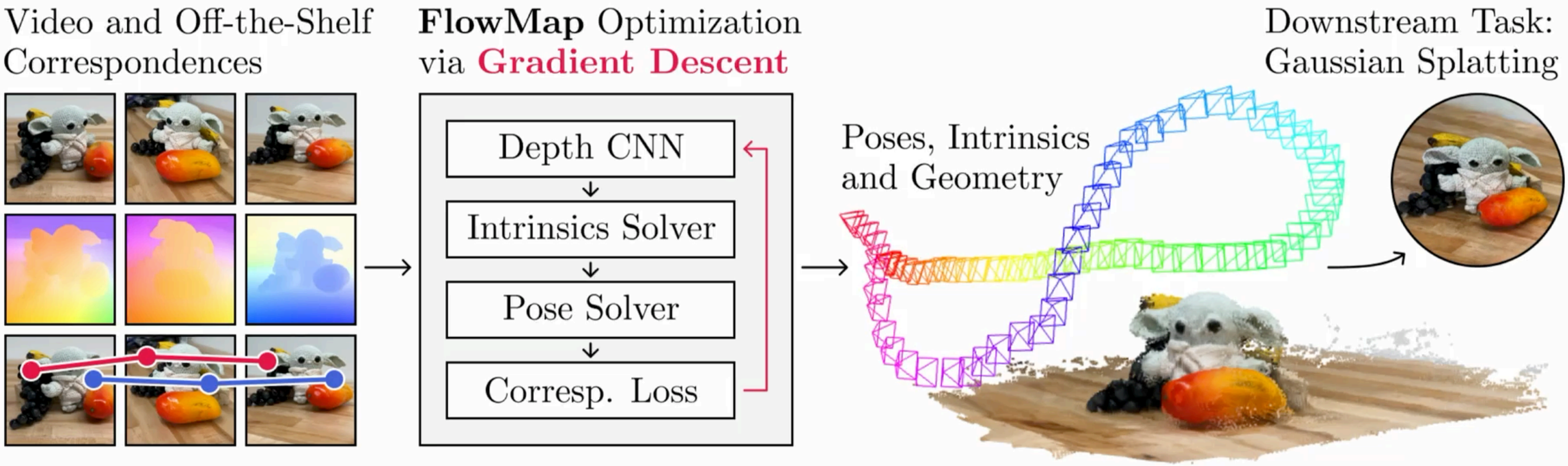
Let's deeply understand that relationship.

on,

Gengshan Yang et al, CVPR 2020

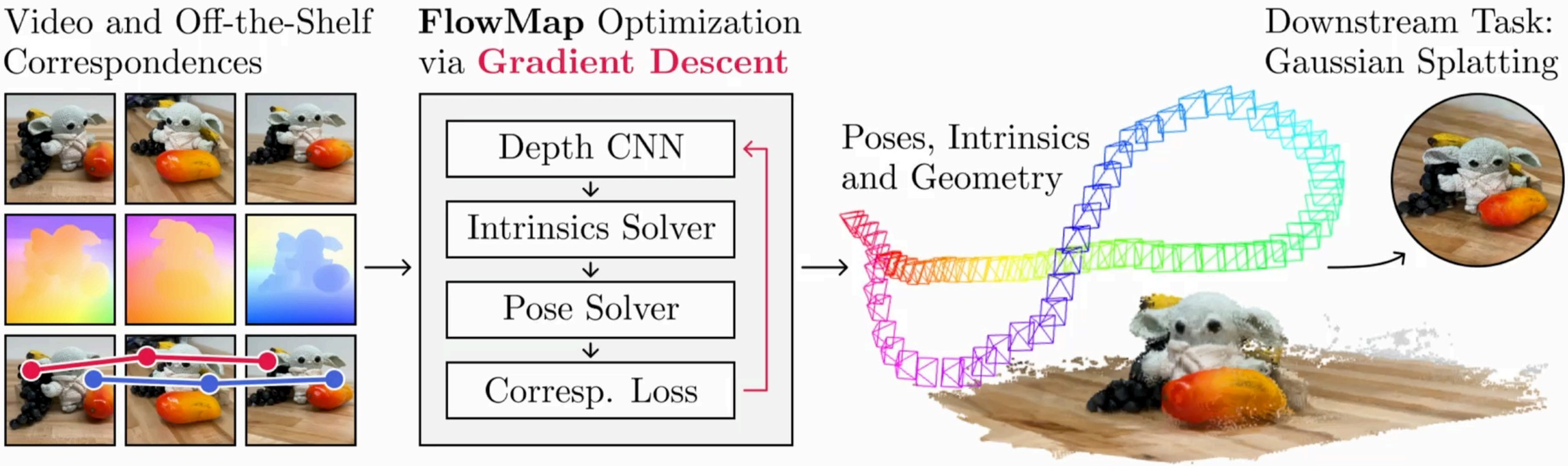
Observation: 2D flow constrains possible 3D flow (for visible points), and optical flow can be computed given 3D scene flow plus depth

FlowMap: Differentiable Structure-from-Motion via Gradient Descent



Gaussian Splats on par with COLMAP's!

FlowMap: Differentiable Structure-from-Motion via Gradient Descent



Gaussian Splats on par with COLMAP's!

Input/output

- Input : Video + Optical flow + sparse point tracks (i.e. correspondences)
  - From Off-the-Shelf State of the Art (e.g. previous slides!)
  - Makes everything easier because we have known correspondences
  - Optional: initialization depth maps
- Predict depth map + camera parameters for each frame

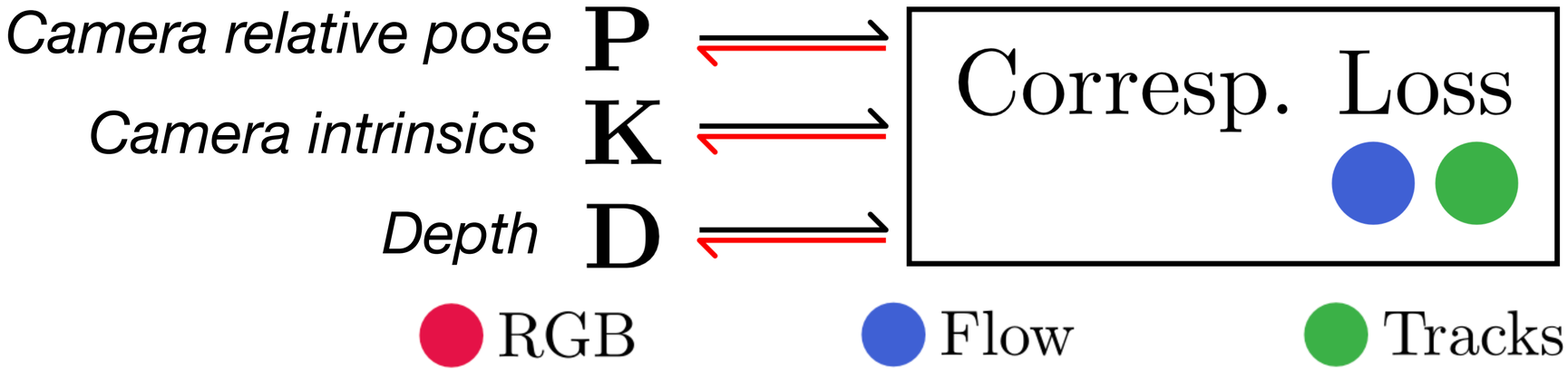


The Camera-Induced Flow Loss

Predict 2D flow induced by relative pose and depth

Compare to input flow/tracks -> loss

Flow and tracks provide redundancy/robustness



RGB Flow Tracks

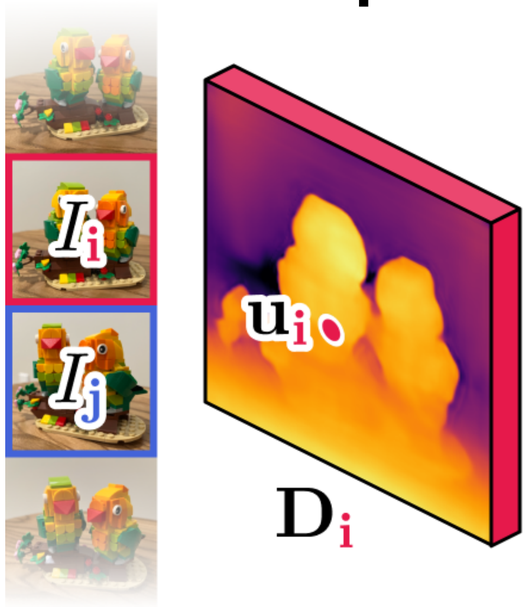


The Camera-Induced Flow Loss

I_i I_j

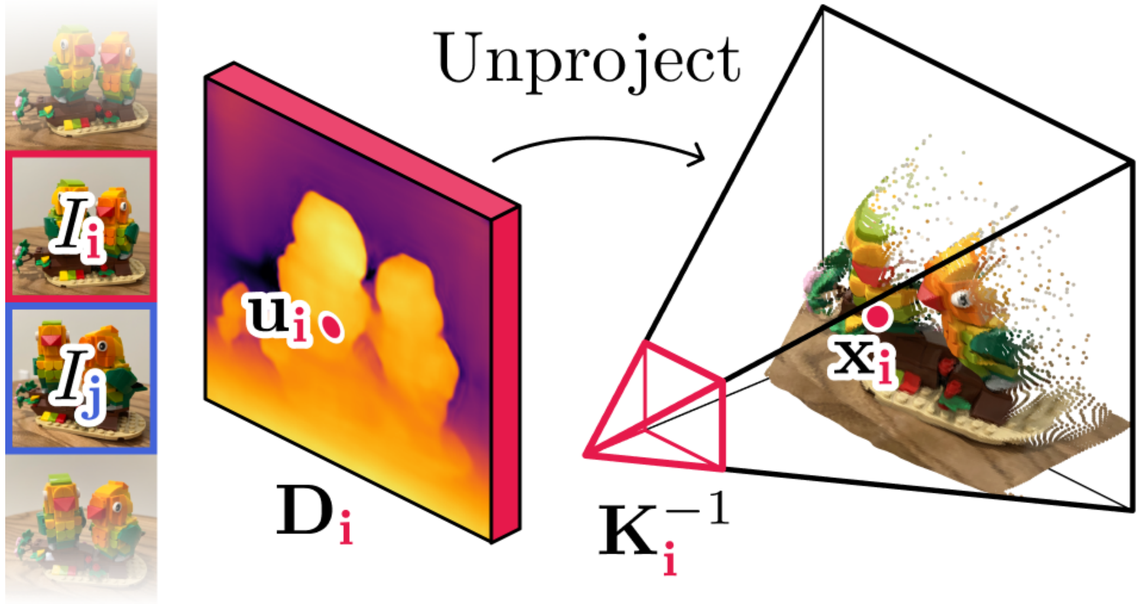


The Camera-Induced Flow Loss



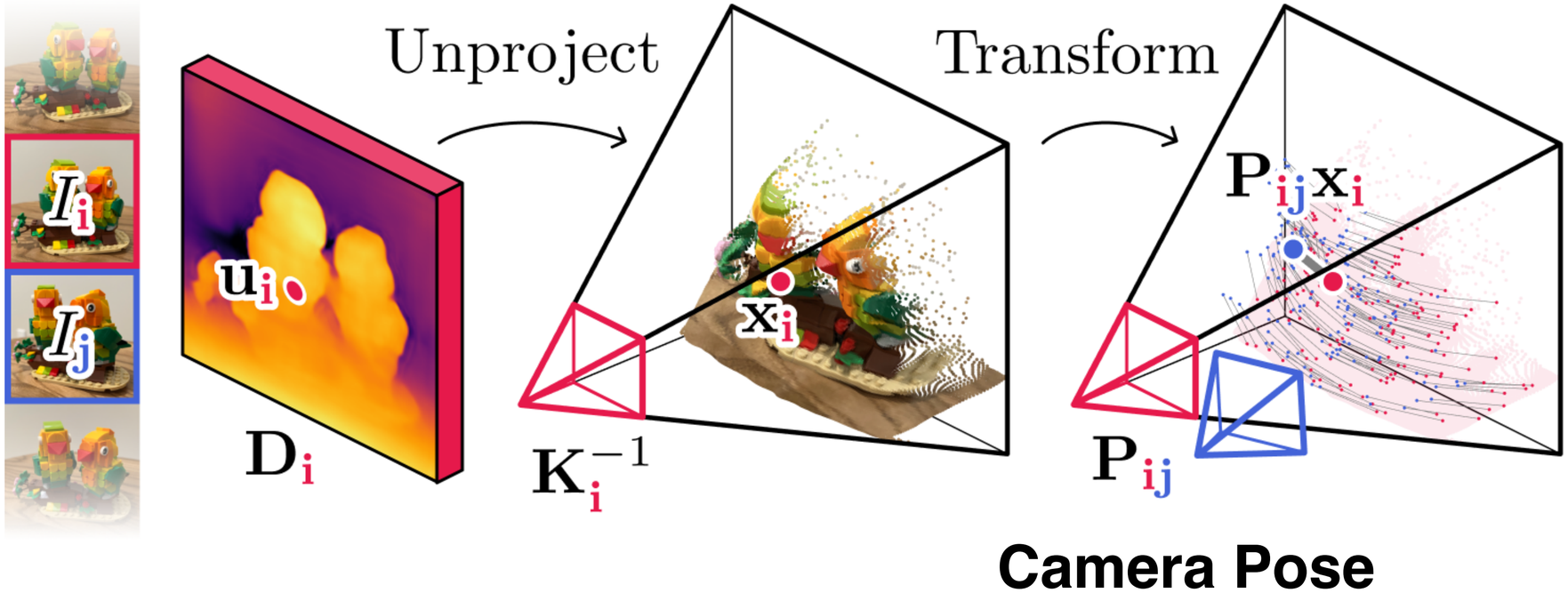
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss



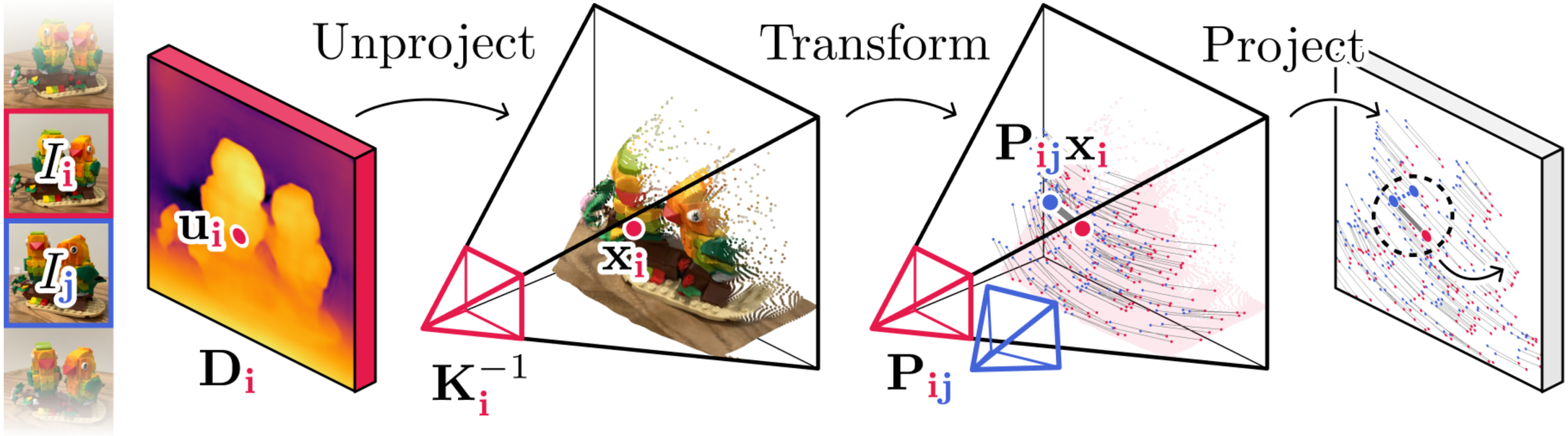
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss



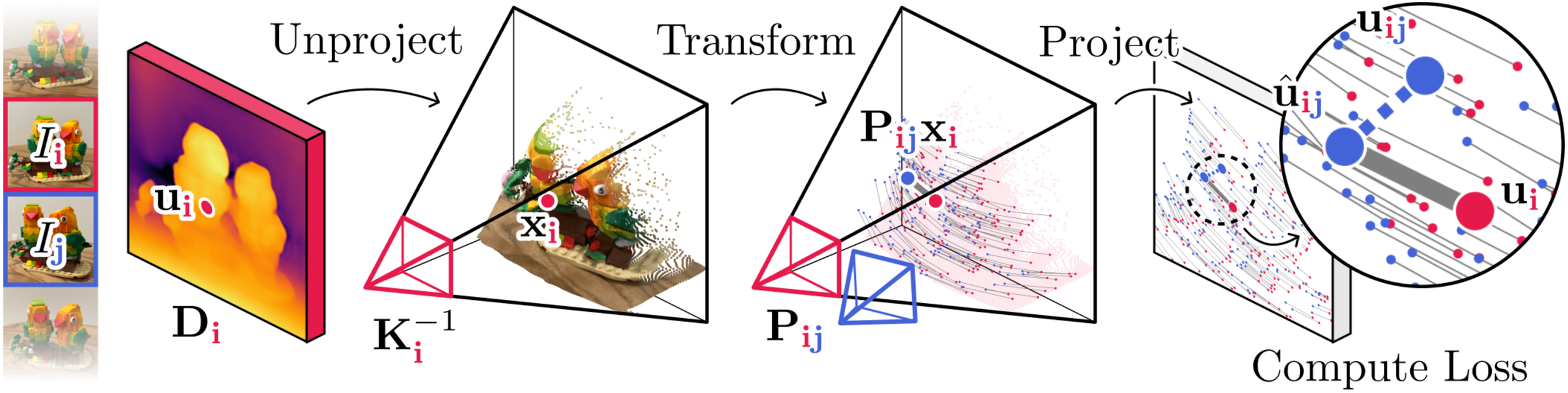
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss

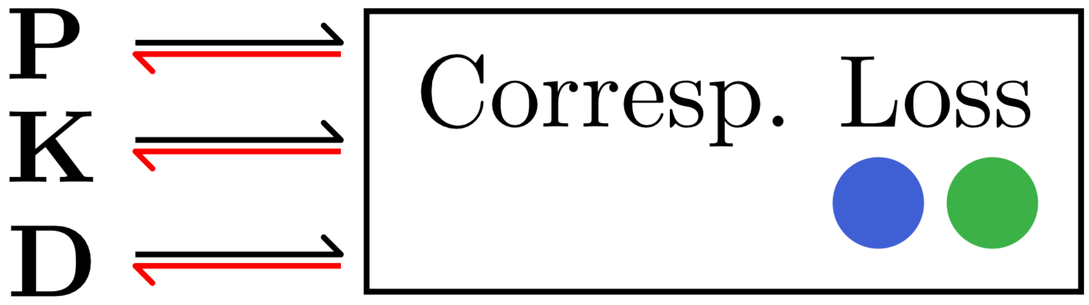


Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss



Let’s optimize depth, pose and intrinsics directly!



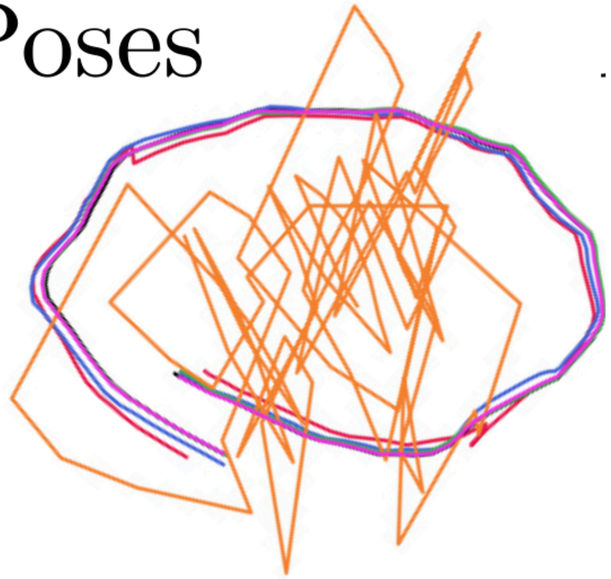


Doesn’t work :/

Pose Depth Focal

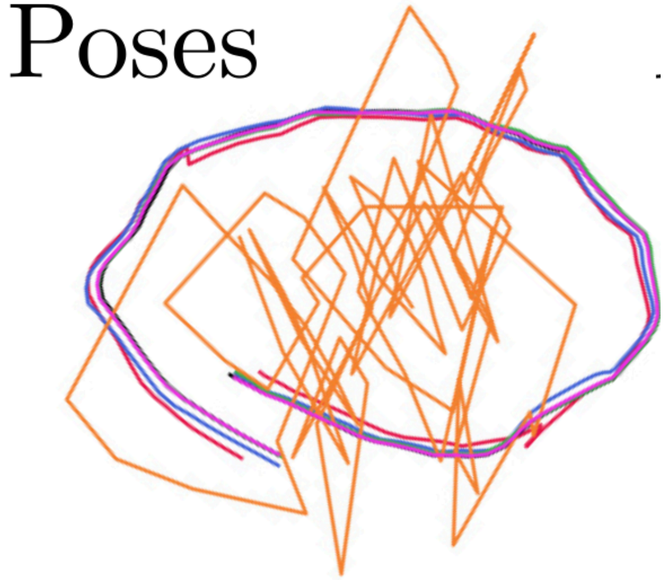


Poses



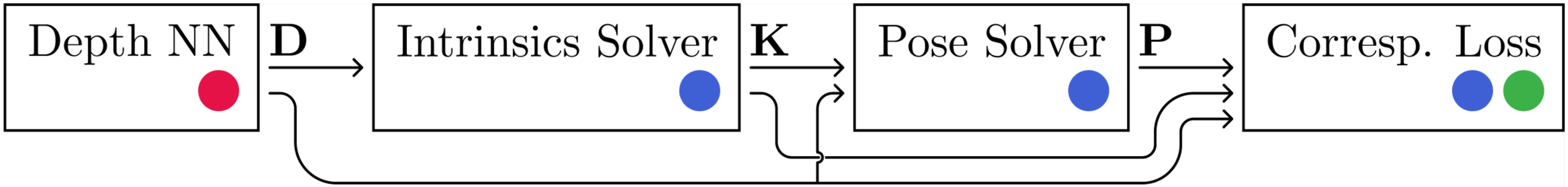
Doesn’t work :/





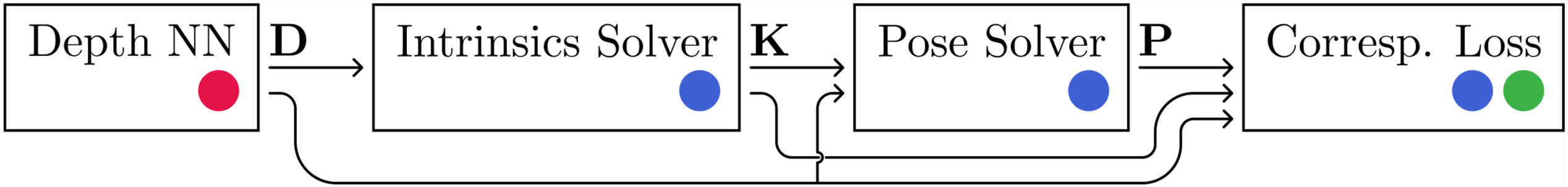
Why not? Lots of local minima: Hard to “get unstuck” from a wrong pose or wrong intrinsics, plus lots of ways for model to “cheat”

FlowMap: Feed-Forward Estimators!





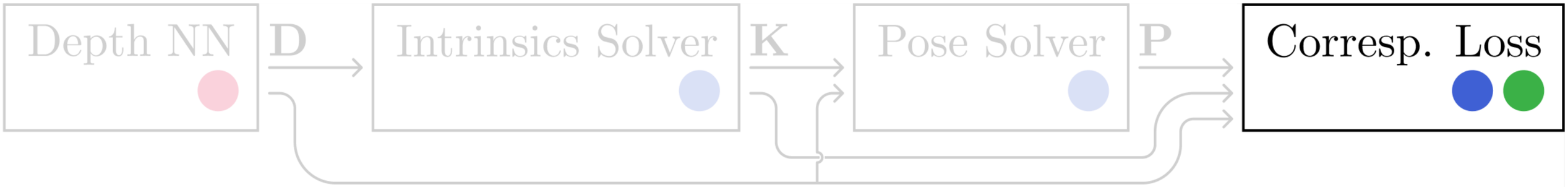
FlowMap: Differentiable SfM via Gradient Descent



RGB Flow Tracks

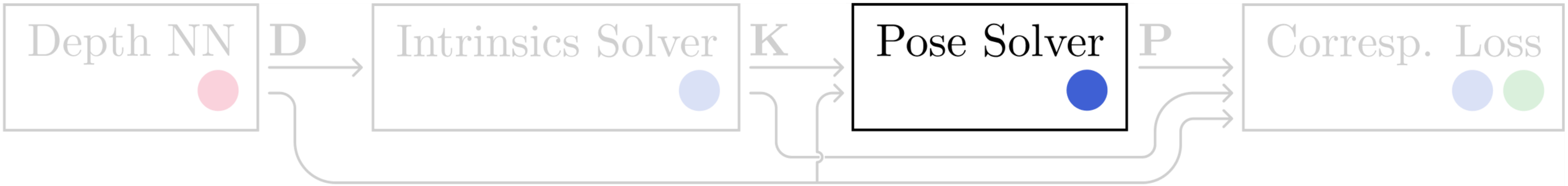


FlowMap: Differentiable SfM via Gradient Descent

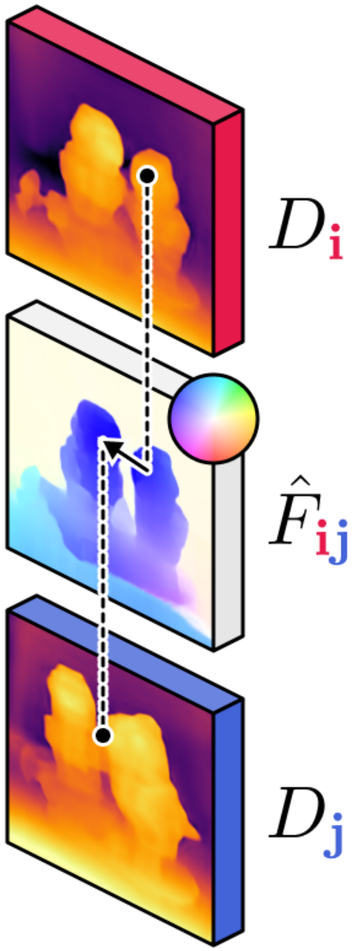


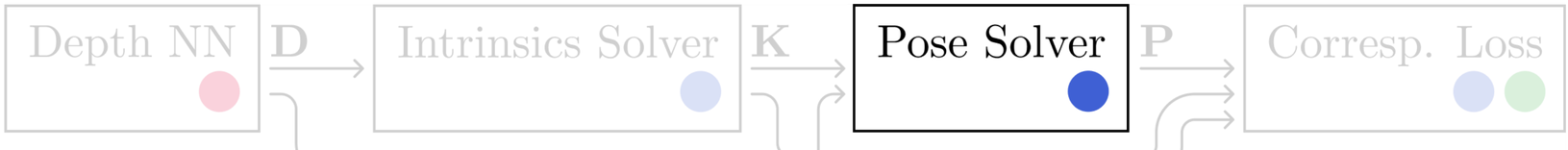
RGB Flow Tracks

FlowMap: Differentiable SfM via Gradient Descent

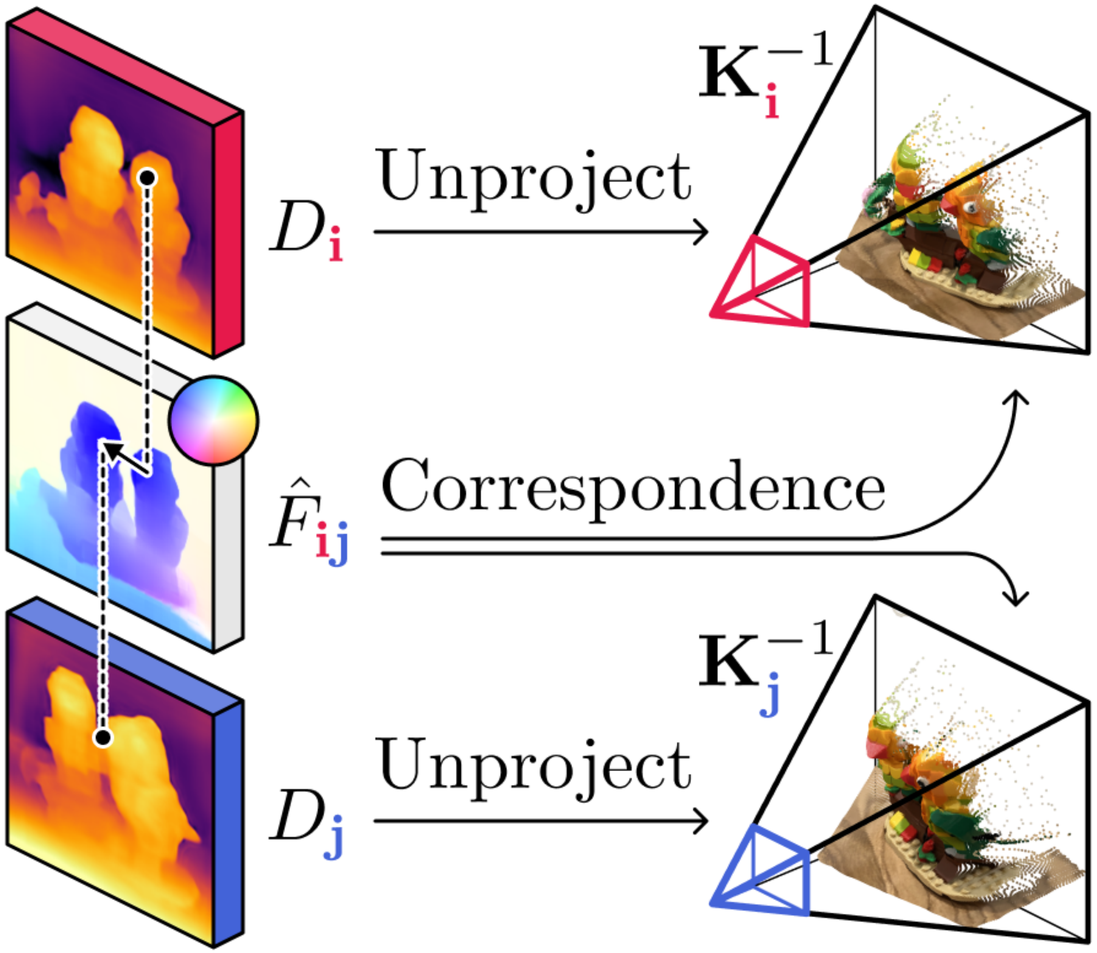


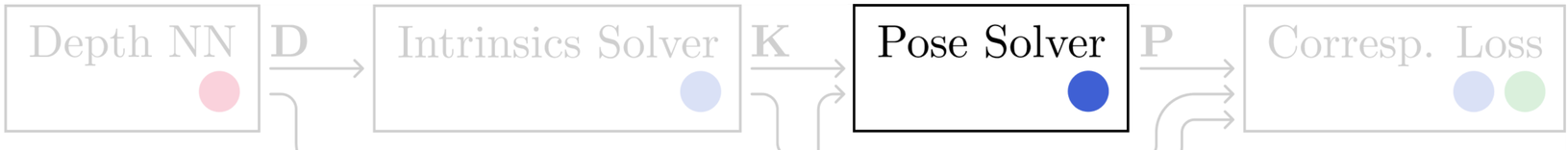
Diff. Least-Squares Pose Solve



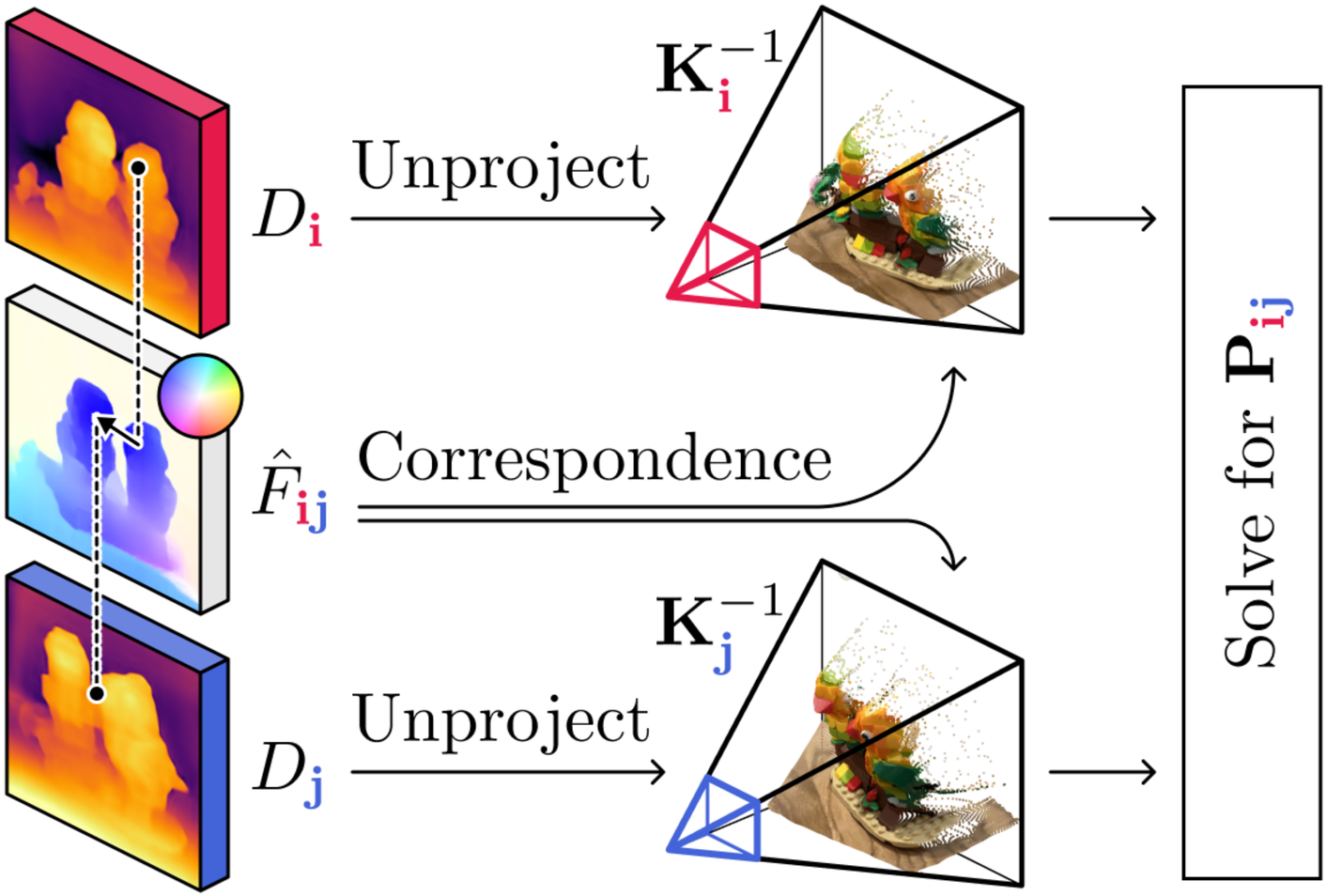


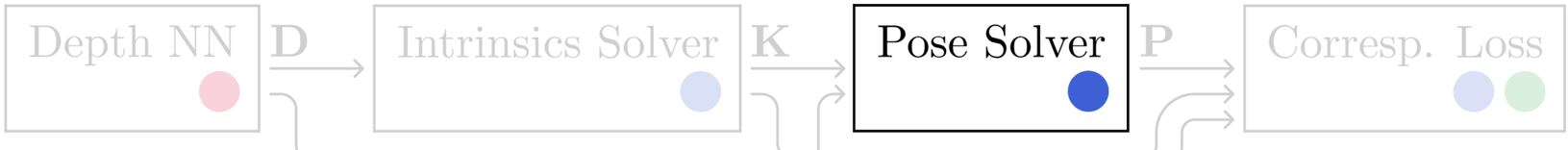
Diff. Least-Squares Pose Solve



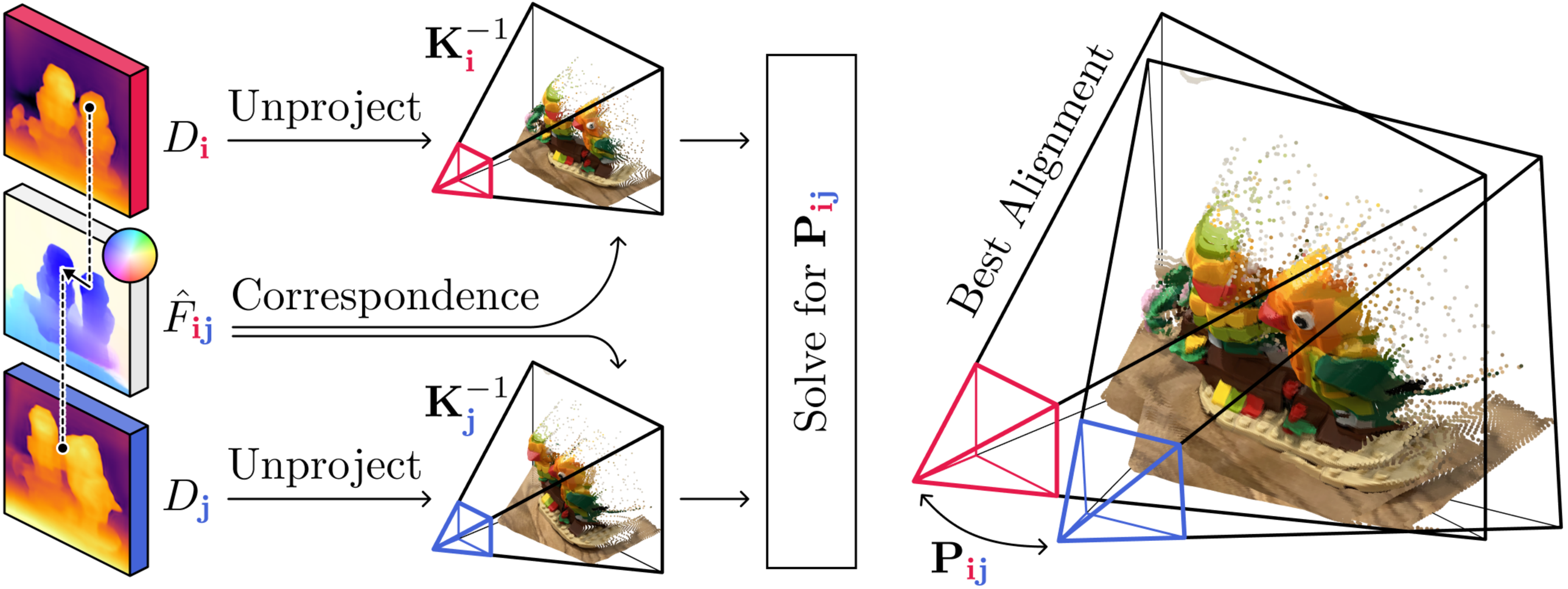


Diff. Least-Squares Pose Solve

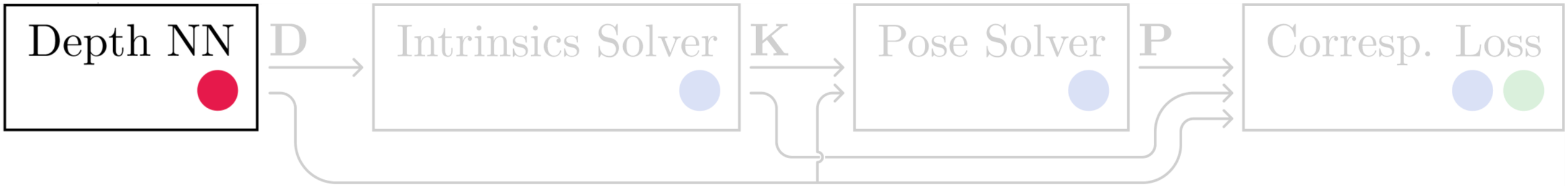




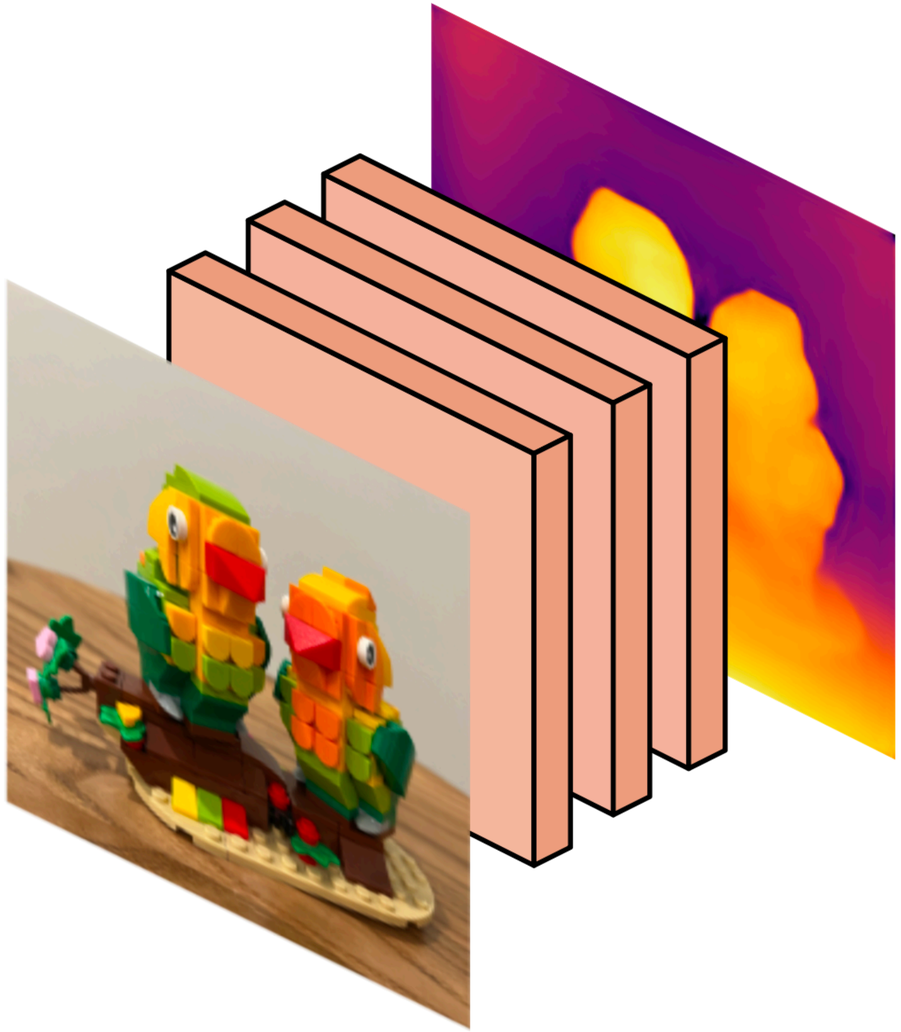
Diff. Least-Squares Pose Solve



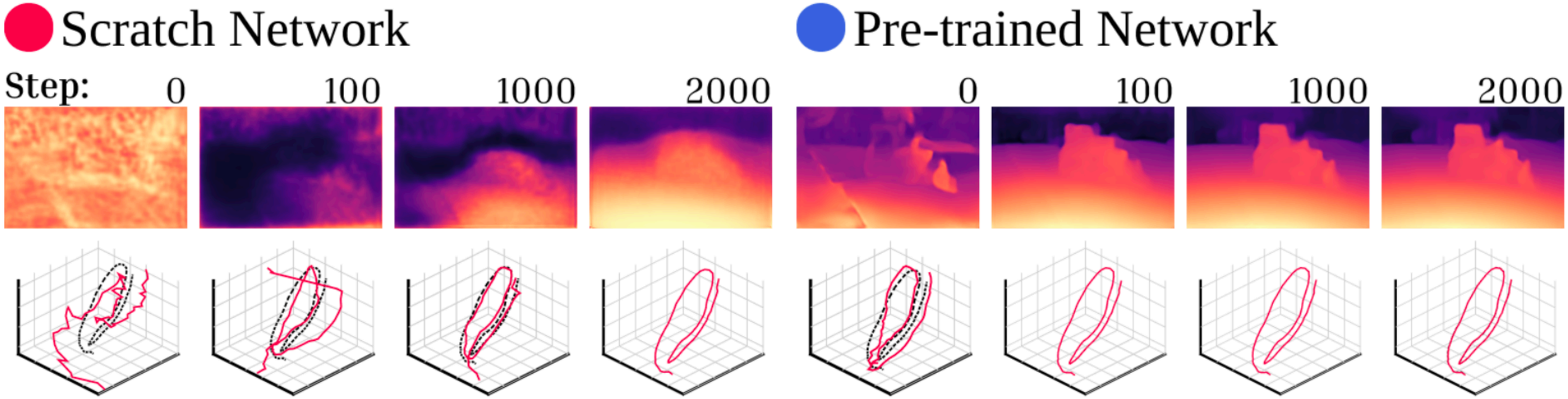
Predict Depth from RGB



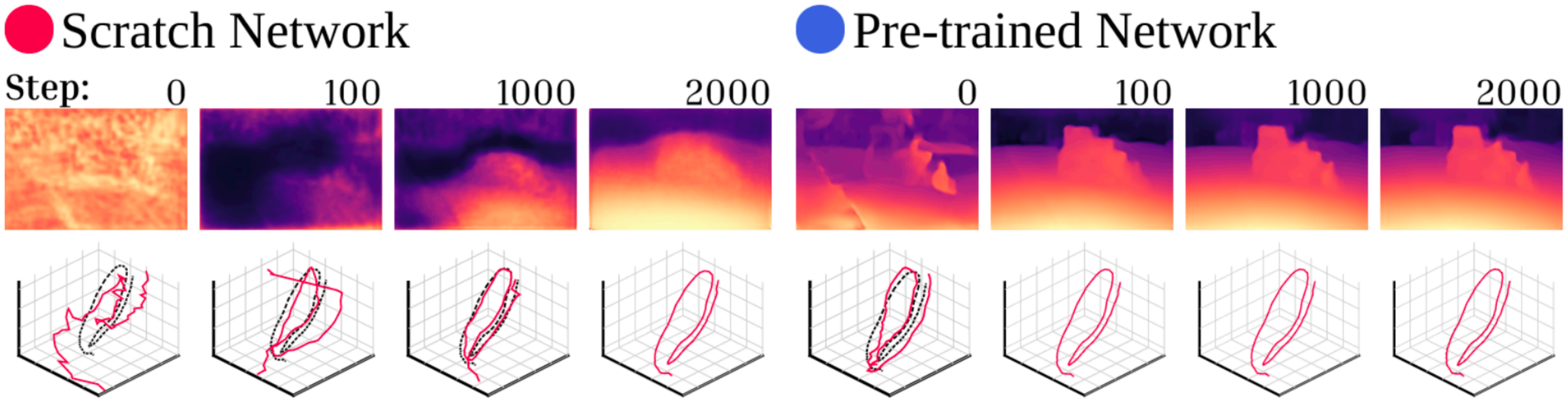
Depth Network



Depth Network Need Not be Pre-trained!

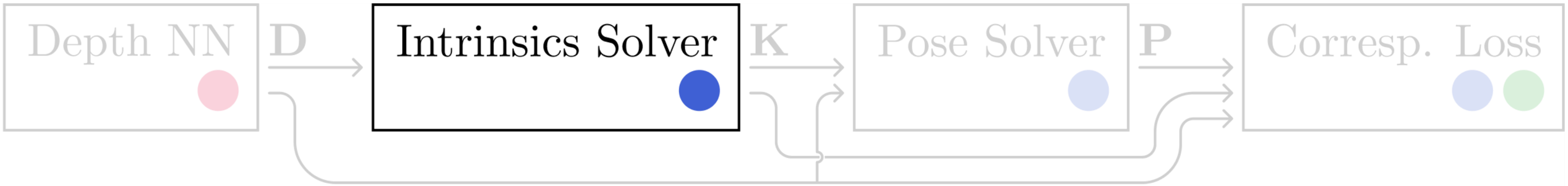


Depth Network Need Not be Pre-trained!

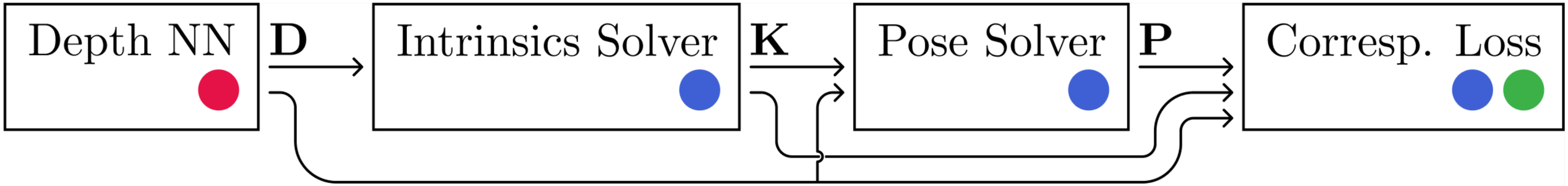


- Depth CNN acts as a **patch-match** regularizer: same RGB patch leads to same depth!
- But: **Can be pre-trained end-to-end on large video datasets without annotations, leading to much faster convergence!**
- In the limit of perfect monocular depth estimator: Feed-Forward SfM!

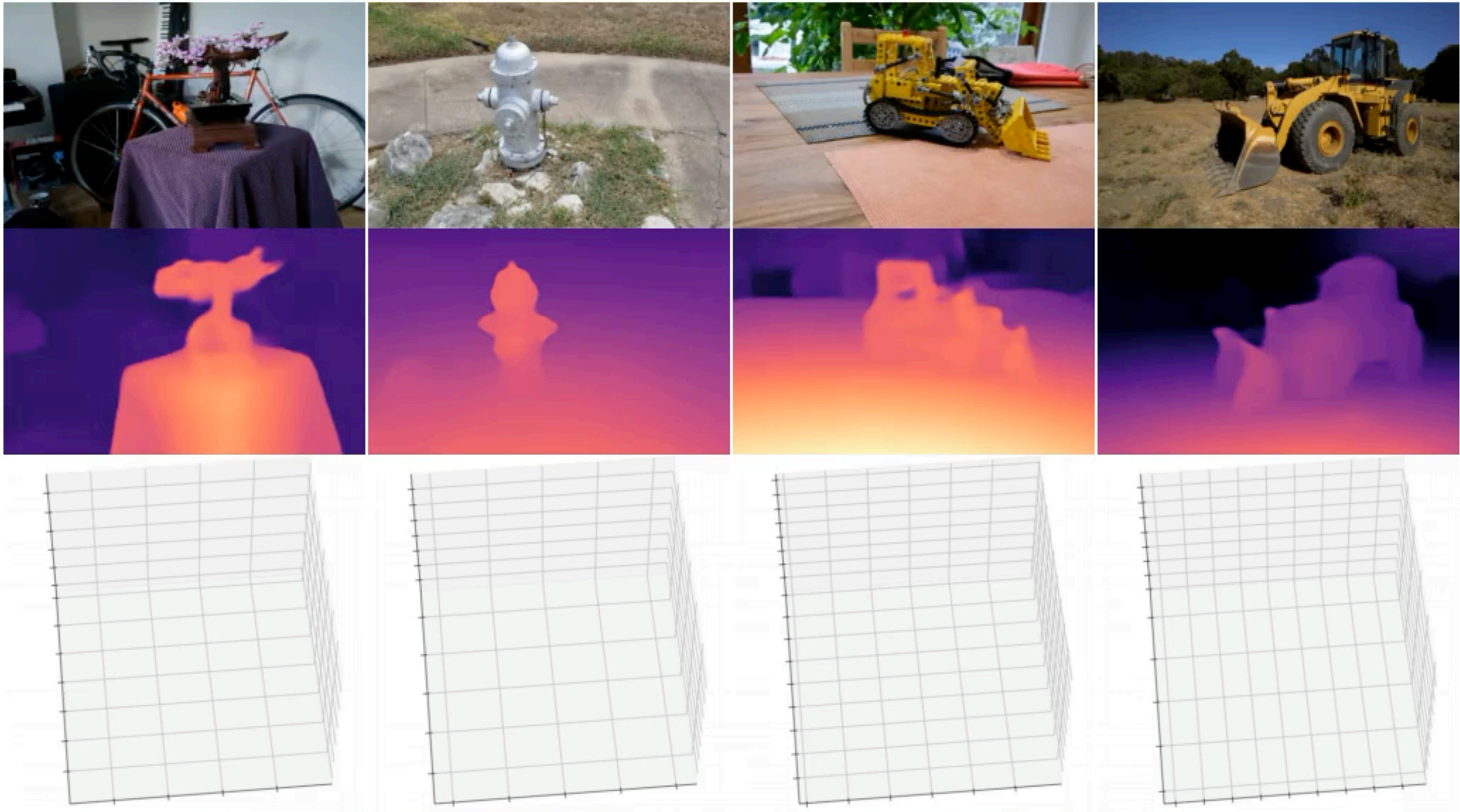
No time to talk about diff. Intrinsics solver - pls look at paper!

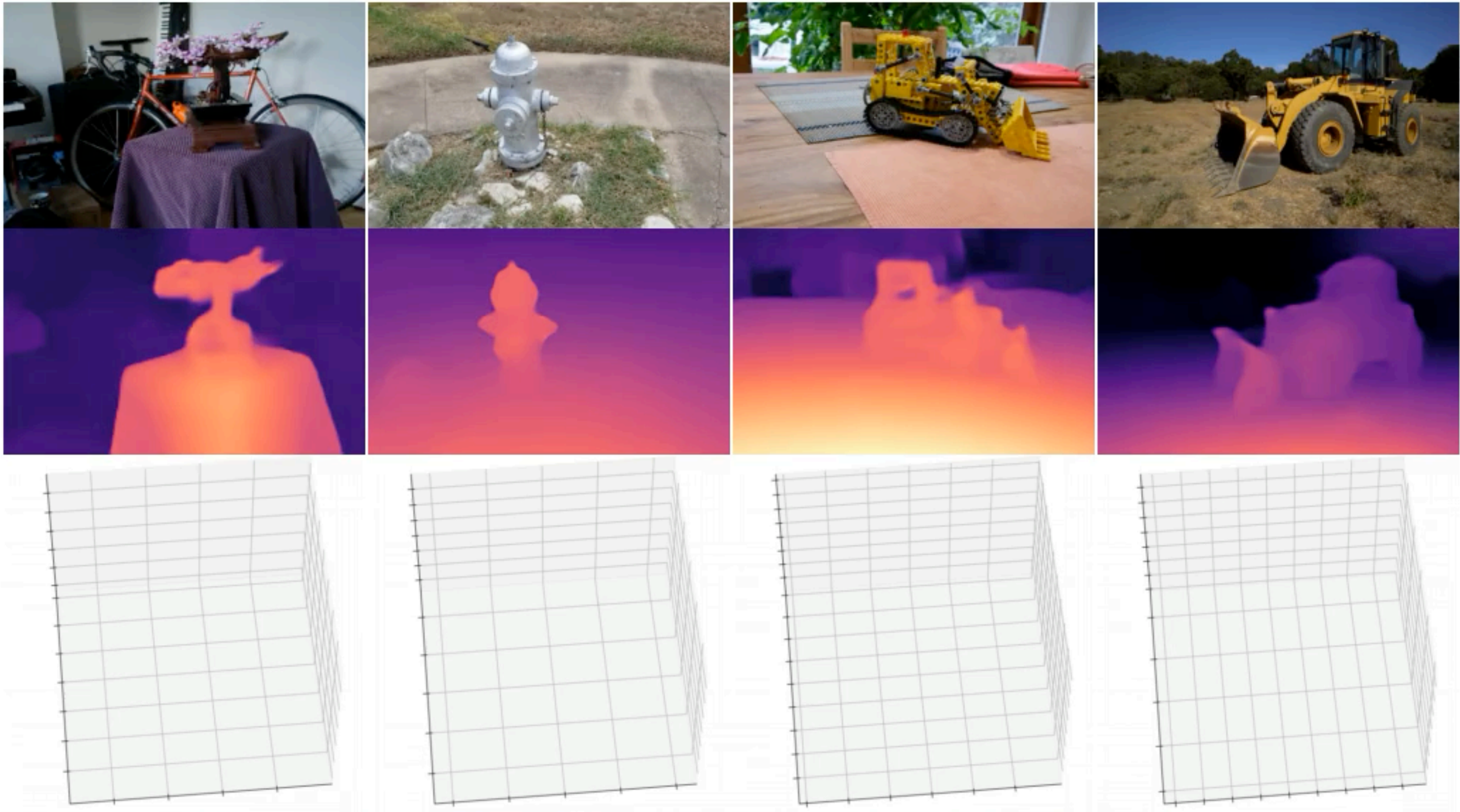


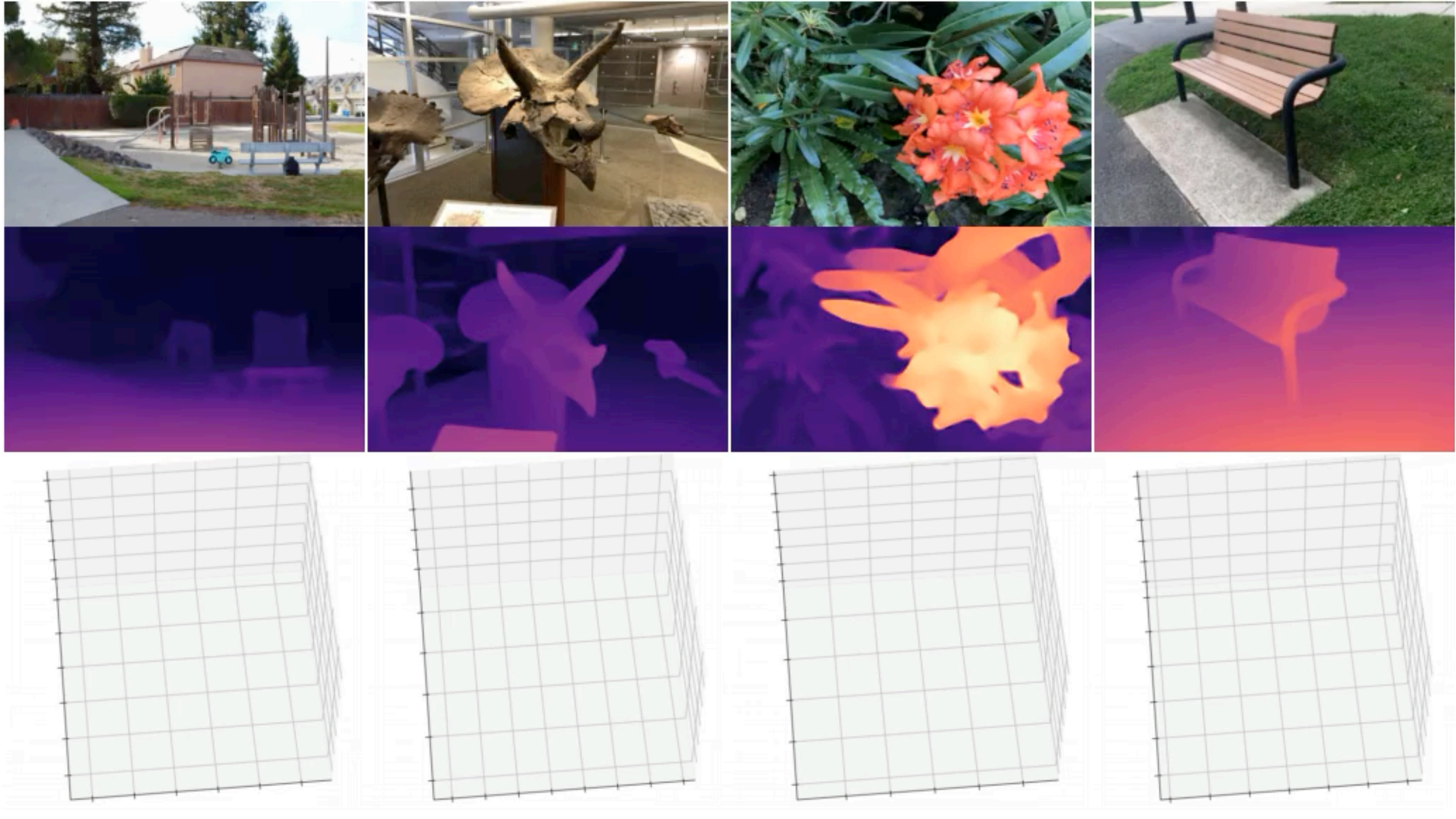
FlowMap: Differentiable SfM via Gradient Descent

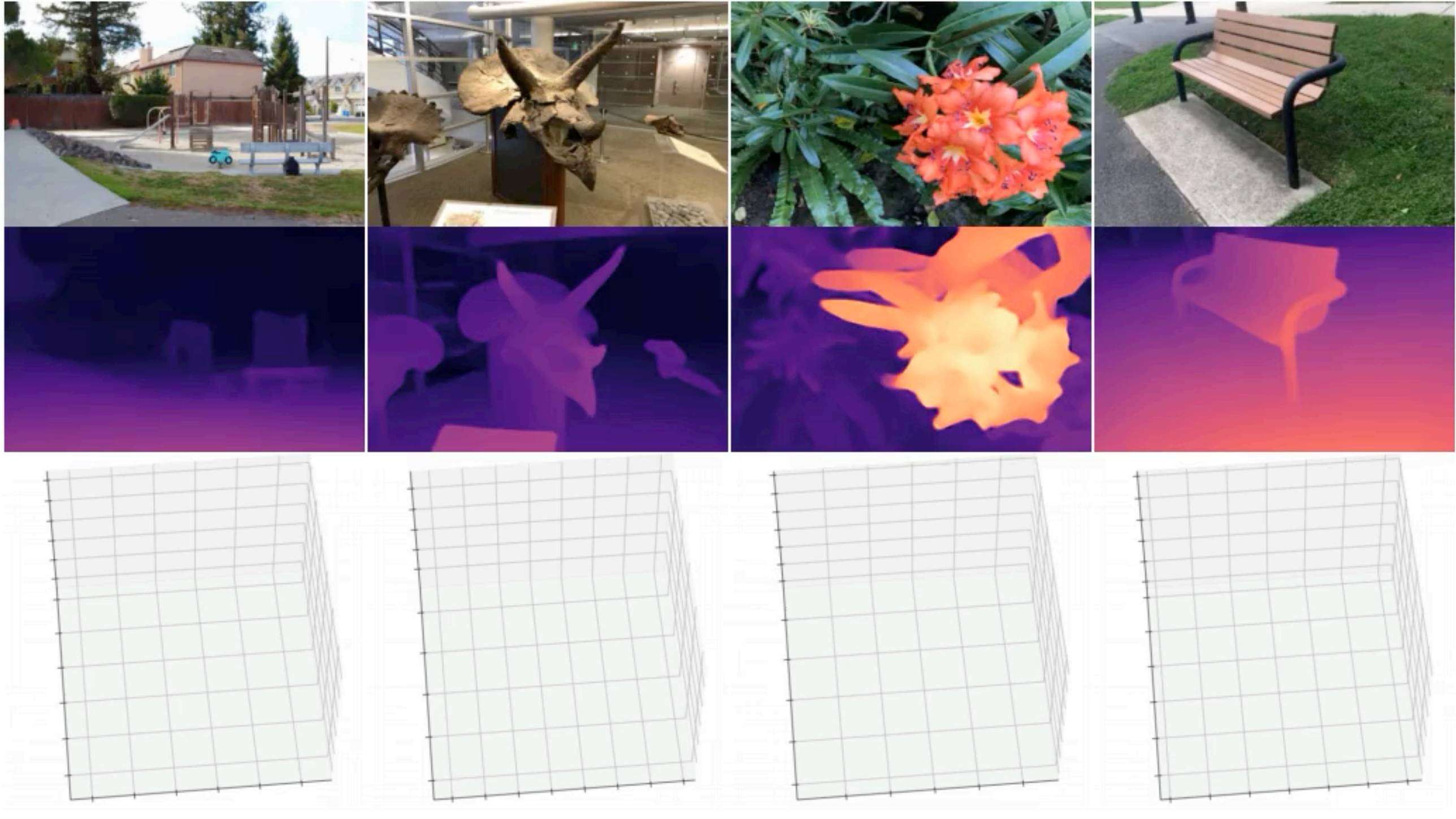


- FlowMap is fit end-to-end via gradient descent on each video separately.
- **Only free parameters are depth network and correspondence weights in pose solver!**
- These weights *can* be pre-trained, but *need not* be pre-trained: The model is constrained almost fully!



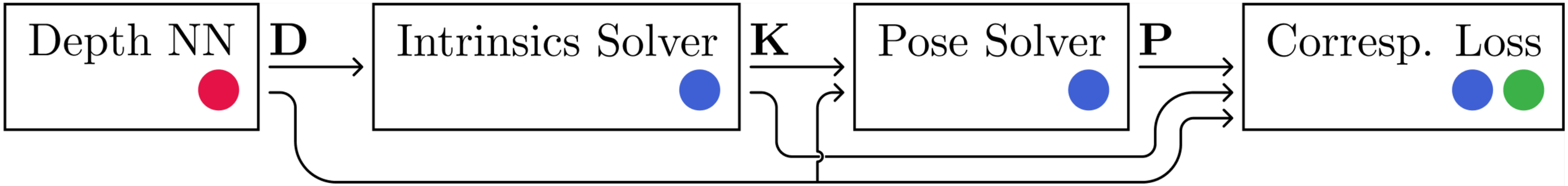








FlowMap: Differentiable SfM via Gradient Descent



- FlowMap is fit end-to-end via gradient descent on each video separately.
- **Only free parameters are depth network and correspondence weights in pose solver!**
- These weights *can* be pre-trained, but *need not* be pre-trained: The model is constrained almost fully!

FlowMap: Differentiable SfM via Gradient Descent



Conclusion: From all you know so far, you can already build a (simple) structure-from-motion model!

- FlowMap is fit end-to-end via gradient descent on each video separately.
- **Only free parameters are depth network and correspondence weights in pose solver!**
- These weights *can* be pre-trained, but *need not* be pre-trained: The model is constrained almost fully!

FlowMap: Differentiable SfM via Gradient Descent



Conclusion: From all you know so far, you can already build a (simple) structure-from-motion model!

More on structure from motion in the next lecture…

-
- **se solver!**
- These weights *can* be pre-trained, but *need not* be pre-trained: The model is constrained almost fully!

Families of correspondence problem(s)

	No 3D, motion	Static 3D
Dense	optical flow	Stereo
Sparse and smooth (+visibility)	Motion tracking	SLAM
Sparse and far	Panorama stitching	Structure From Motion

3D Scene Flow

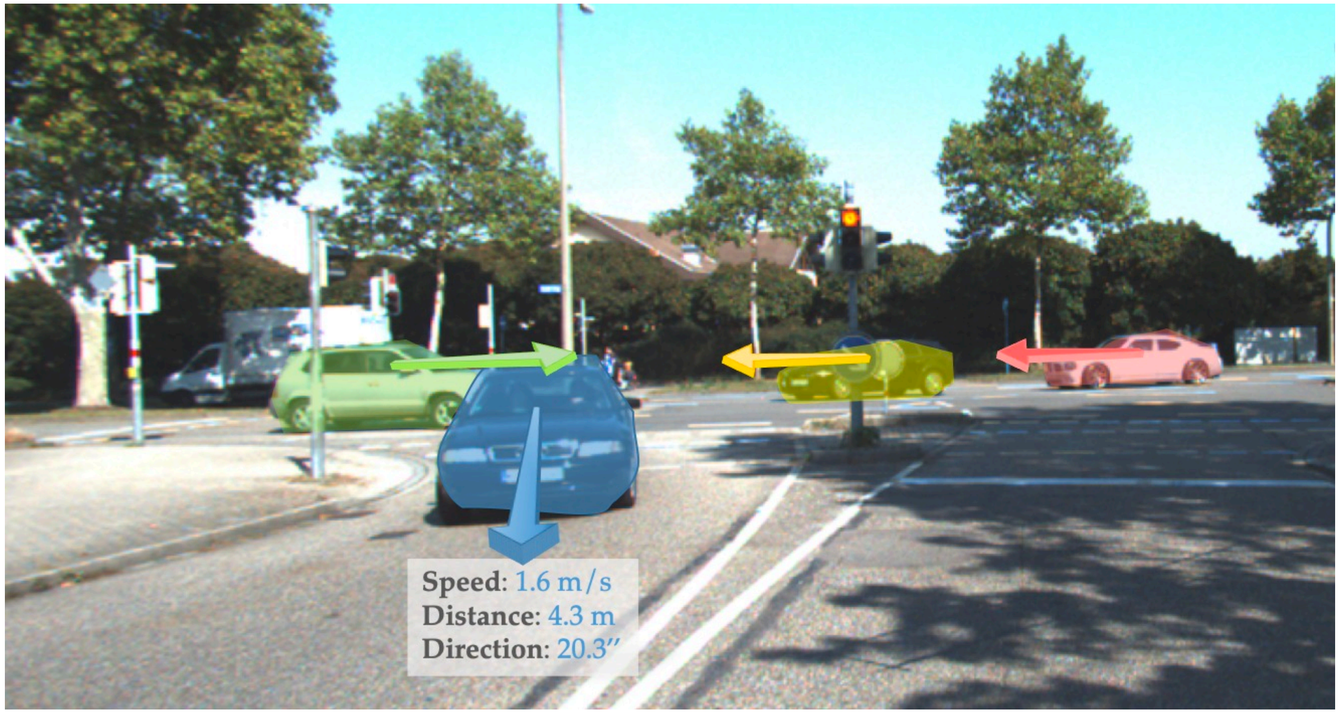
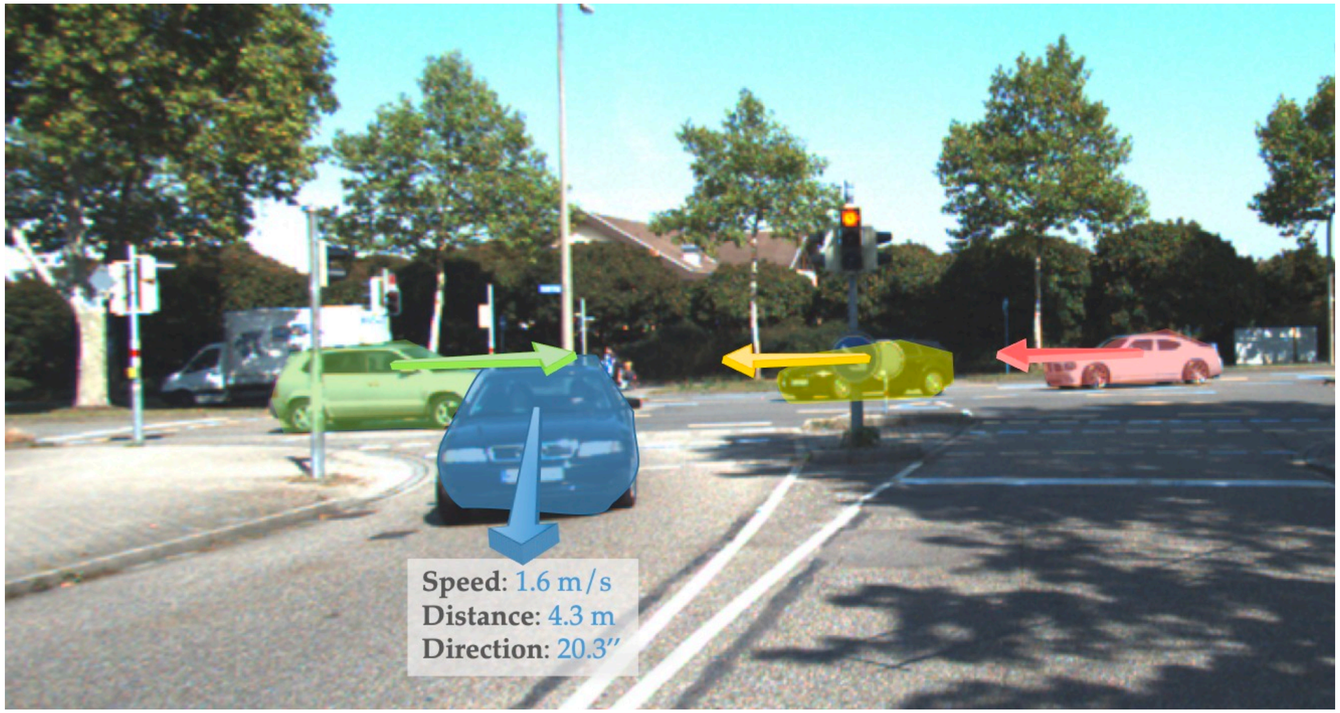


Image Courtesy: Zhe Cao

3D Scene Flow



Observation: 3D flow typically comprised of piecewise rigid motions

Image Courtesy: Zhe Cao

Parametrizing Scene Flow Fields



Image Courtesy: Zhe Cao

Parametrizing Scene Flow Fields



For an object rotating around its centre:

$$\mathbf{x}_{t+1} = R(\mathbf{x}_t - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$$

Image Courtesy: Zhe Cao

Parametrizing Scene Flow Fields



For an object rotating around its centre:

$$\mathbf{x}_{t+1} = R(\mathbf{x}_t - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$$

Option 1: Predict a per-point translation

(requires different predictions for different points)

$$\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2; \quad \mathbf{\Phi}(\mathbf{x}) = \Delta\mathbf{x}$$

Image Courtesy: Zhe Cao

Parametrizing Scene Flow Fields



For an object rotating around its centre:

$$\mathbf{x}_{t+1} = R(\mathbf{x}_t - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$$

Option 1: Predict a per-point translation

(requires different predictions for different points)

$$\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}^2; \quad \mathbf{\Phi}(\mathbf{x}) = \Delta\mathbf{x}$$

Image Courtesy: Zhe Cao

Option 2: Predict a per-point translation **and** rotation

(allows **same** prediction for **all** points within a rigid object)

$$\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^2 \rightarrow SE(3); \quad \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$$

Predicting 3D Scene Flow (supervised)

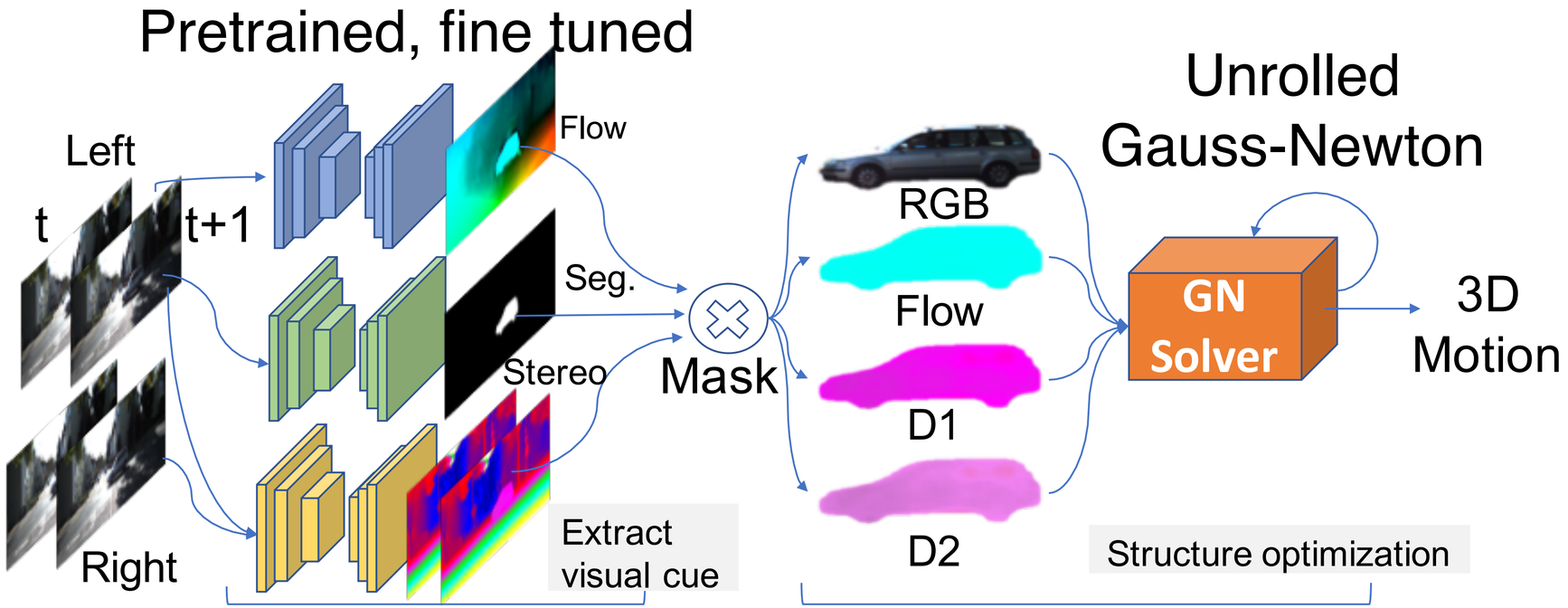
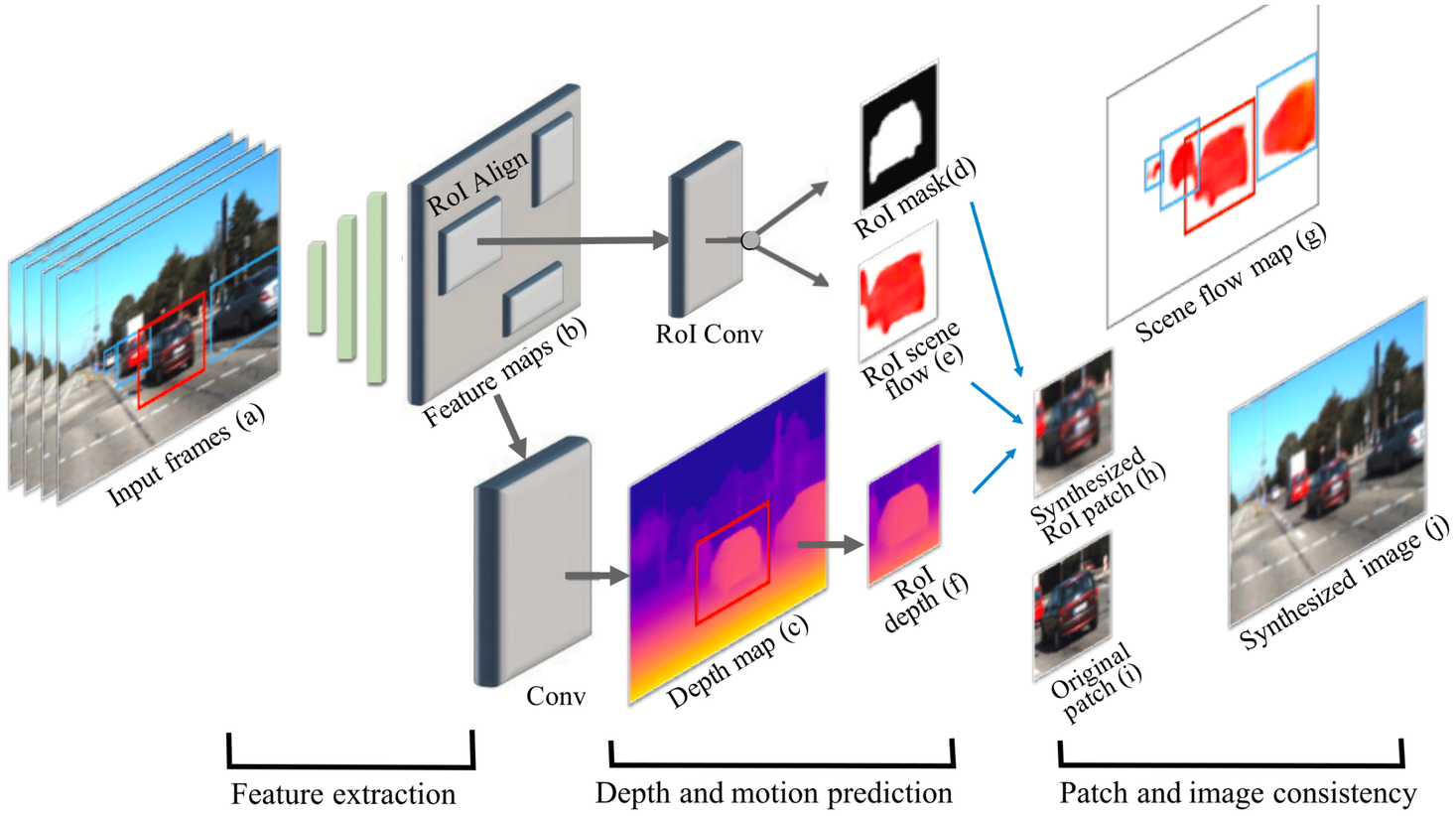


Figure 2: **Overview of our approach:** Given two consecutive stereo images, we first estimate the flow, stereo, and segmentation (Sec. 3.1). The visual cues of each instance are then encoded as energy functions (Sec. 3.2) and passed into the Gaussian-Newton (GN) solver to find the best 3D rigid motion (Sec. 3.3). The GN solver is unrolled as a recurrent network.

Estimate a per-instance 3D motion consistent with predicted depths and flow

Image Courtesy: Deep Rigid Instance Scene Flow. Ma et. al.

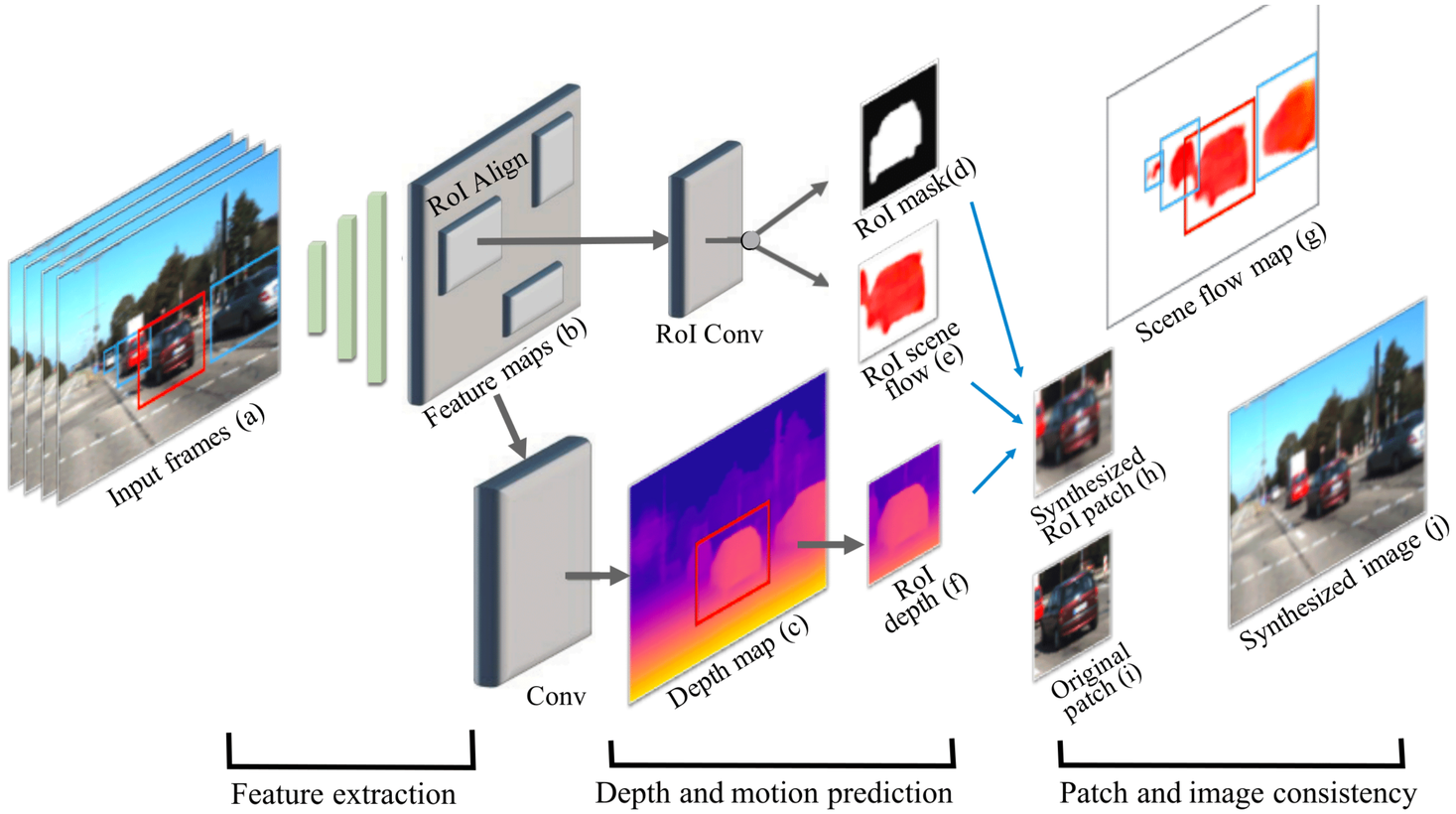
Predicting 3D Scene Flow from stereo pairs (less supervised)



https://arxiv.org/pdf/1901.01971

Image Courtesy: Learning Independent Object Motion from Unlabelled Stereoscopic Videos. Cao et. al.

Predicting 3D Scene Flow from stereo pairs (less supervised)



Self-supervised learning of depth and scene flow

https://arxiv.org/pdf/1901.01971

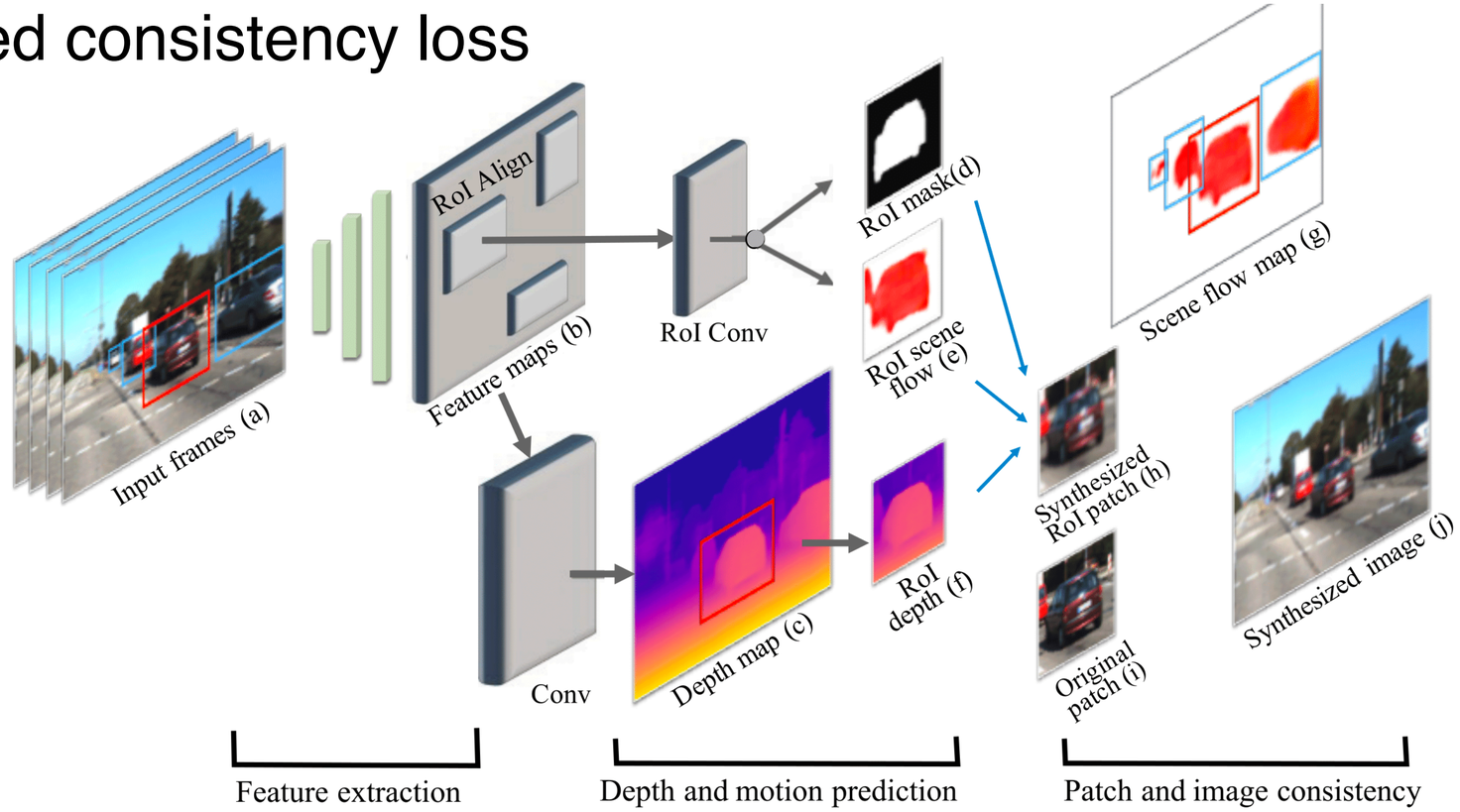
Image Courtesy: Learning Independent Object Motion from Unlabelled Stereoscopic Videos. Cao et. al.

Predicting 3D Scene Flow from stereo pairs (less supervised)

Only bounding boxes needed at training

Rigidity not assumed

Warping—based consistency loss



Self-supervised learning of depth and scene flow

https://arxiv.org/pdf/1901.01971

Image Courtesy: Learning Independent Object Motion from Unlabelled Stereoscopic Videos. Cao et. al.

Self-Supervised Monocular Scene Flow Estimation

Junhwa Hur Stefan Roth
Department of Computer Science, TU Darmstadt



Figure 1. **Results of our monocular scene flow approach on the KITTI dataset [11].** Given two consecutive images *(left)*, our method jointly predicts depth *(middle)* and scene flow *(right)*. (x,z)-coordinates of 3D scene flow are visualized using an optical flow color coding.

More challenging because no stereo, scale ambiguity

View synthesis (~warping) loss

Based on optical flow architecture

Decompose cost volume into 3D scene flow + depth

Self-supervised learning of depth and scene flow

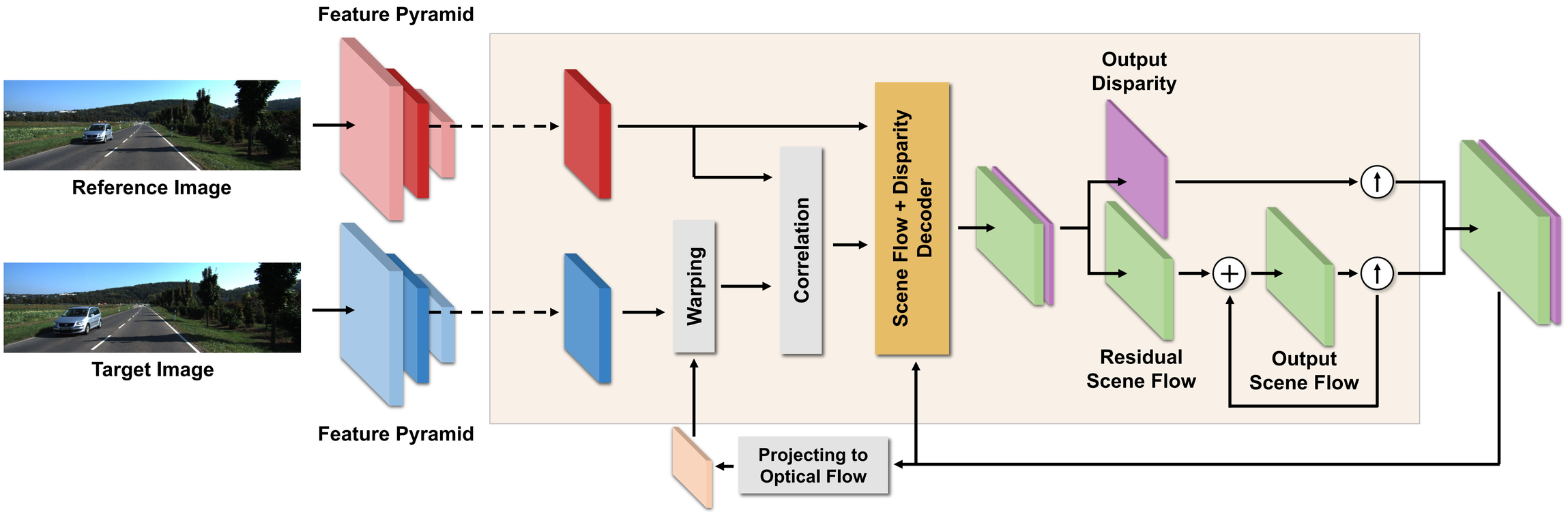


Image Courtesy: Self-Supervised Monocular Scene Flow Estimation. Hur et. al.

Self-supervised learning of depth and scene flow

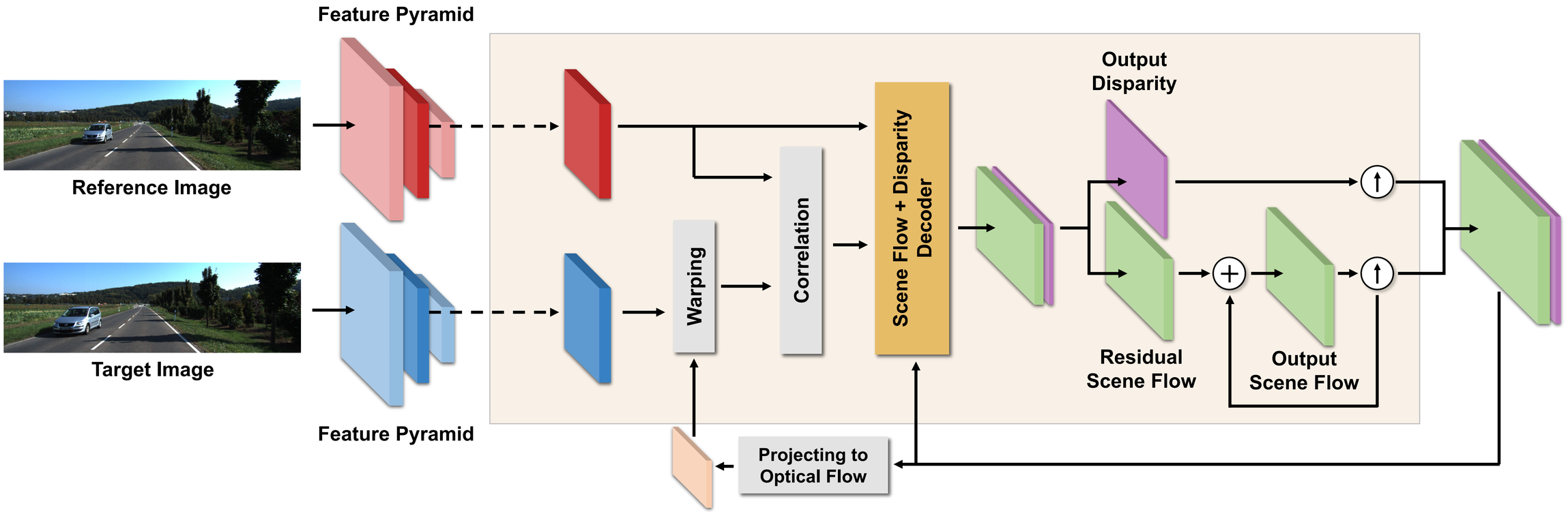
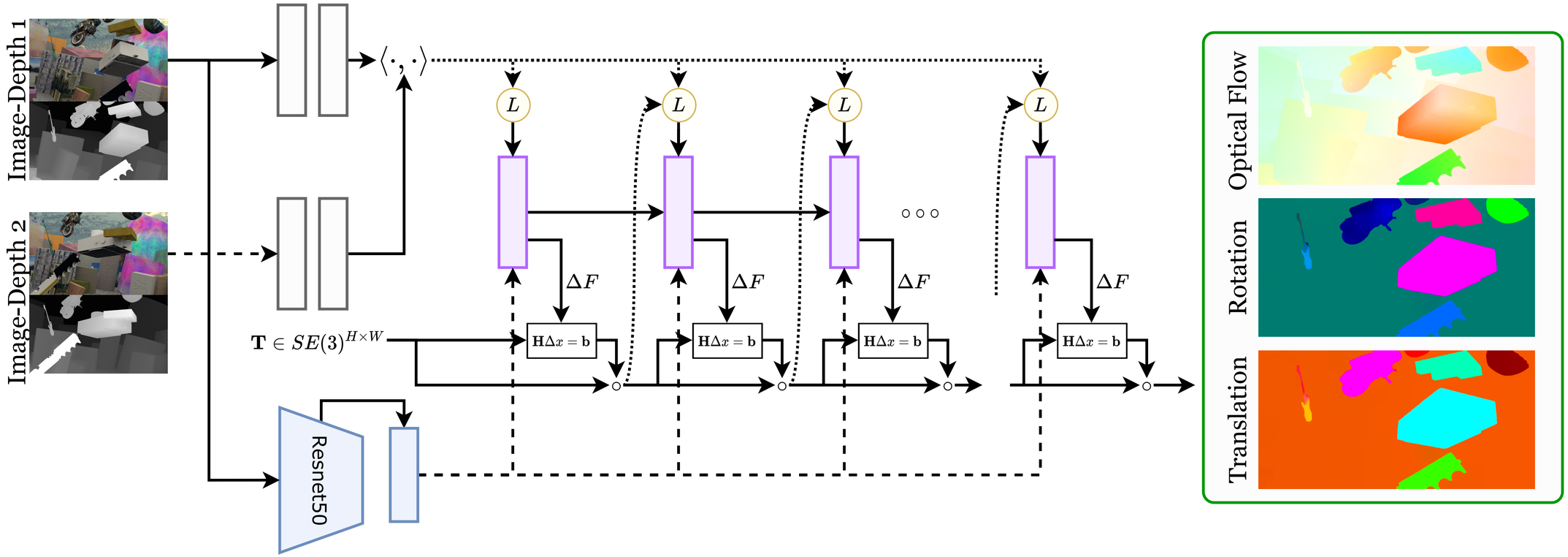


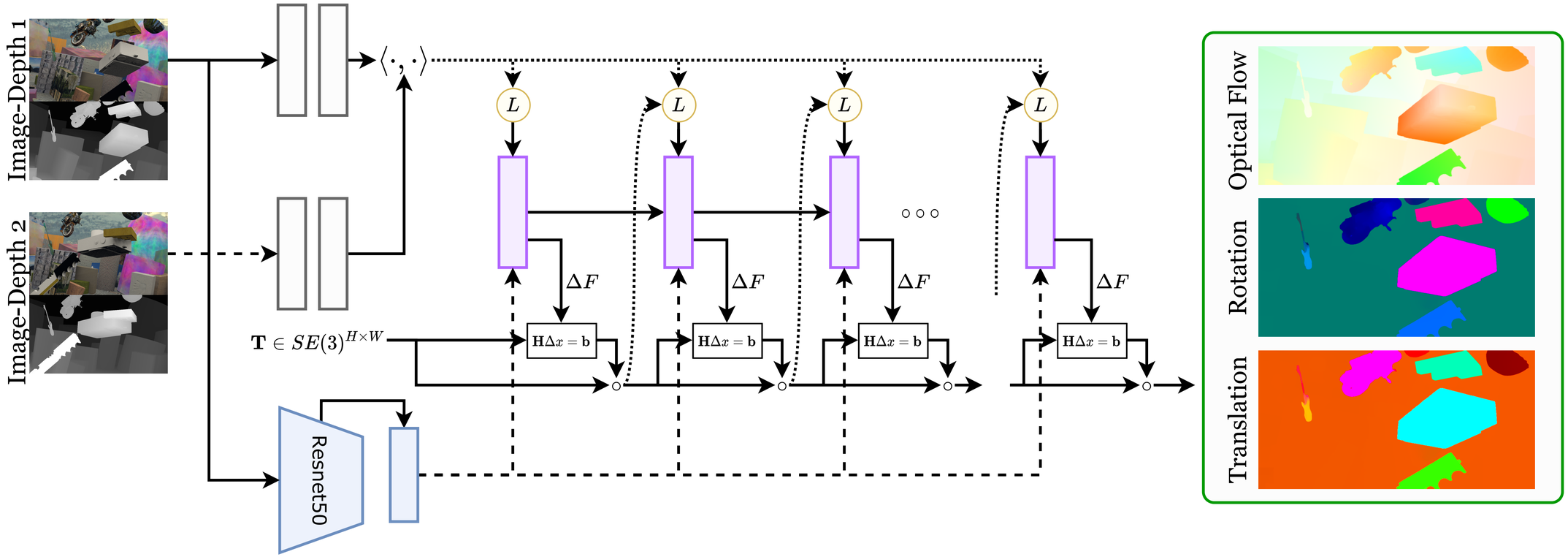
Image Courtesy: Self-Supervised Monocular Scene Flow Estimation. Hur et. al.

Predicting 3D Scene Flow



RAFT-3D: Scene Flow using Rigid-Motion Embeddings. Teed et. al.

Predicting 3D Scene Flow



RAFT-3D: Scene Flow using Rigid-Motion Embeddings. Teed et. al.

Iteratively update pixelwise motion fields in context of a 2D cost volume while enforcing that nearby pixels have a similar motion

A ‘Canonical’ Representation



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

A ‘Canonical’ Representation



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

A ‘Canonical’ Representation

Key idea: Scene at at time is deformation of a “canonical” scene.



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

A ‘Canonical’ Representation



Live Input Depth Map



Live Model Output



Live RGB Image (unused)

Key idea: Scene at at time is deformation of a “canonical” scene.

Represents scene flow as $\Phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow SE(3)$.



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

A ‘Canonical’ Representation



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

Key idea: Scene at at time is deformation of a “canonical” scene.

Represents scene flow as $\Phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow SE(3)$.

This allows straight-forward smoothness regularization!

A ‘Canonical’ Representation



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

(Relative camera motion removed)

Video credits: DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time.
Newcombe et. al

Key idea: Scene at at time is deformation of a “canonical” scene.

Represents scene flow as $\Phi(\mathbf{x}) : \mathbb{R}^3 \rightarrow SE(3)$.

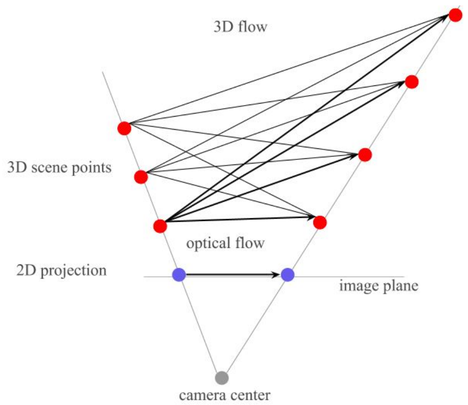
This allows straight-forward smoothness regularization!

Jointly optimizes the time-independent canonical shape and the per-frame warp field.

3D Scene Flow: Some takeaways

3D Scene Flow: Some takeaways

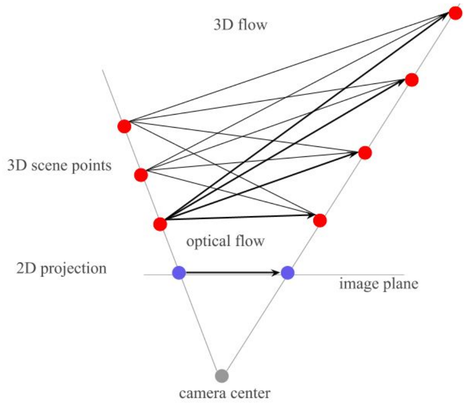
Scene flow allows predicting optical flow



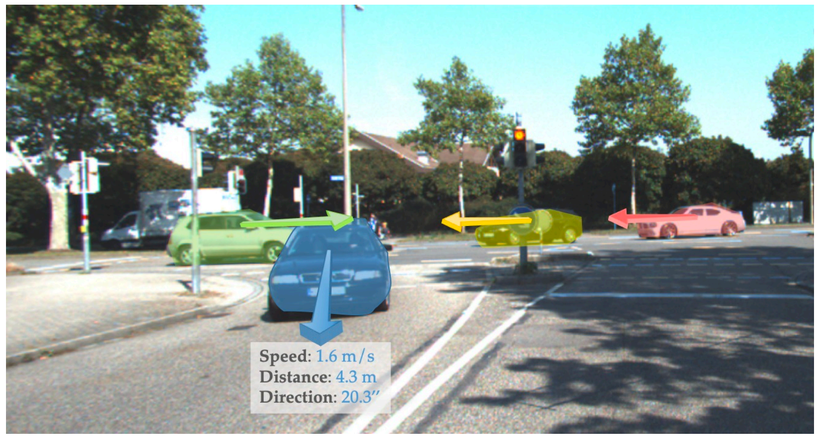
Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

3D Scene Flow: Some takeaways

Scene flow allows predicting optical flow



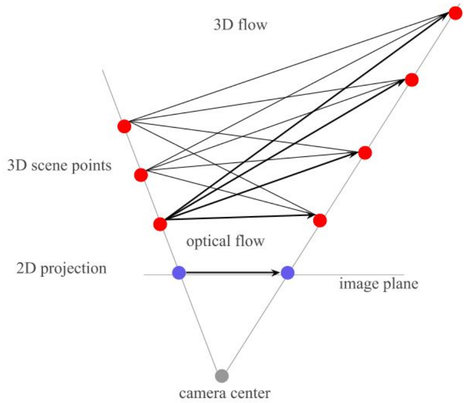
Typical motions are piecewise rigid



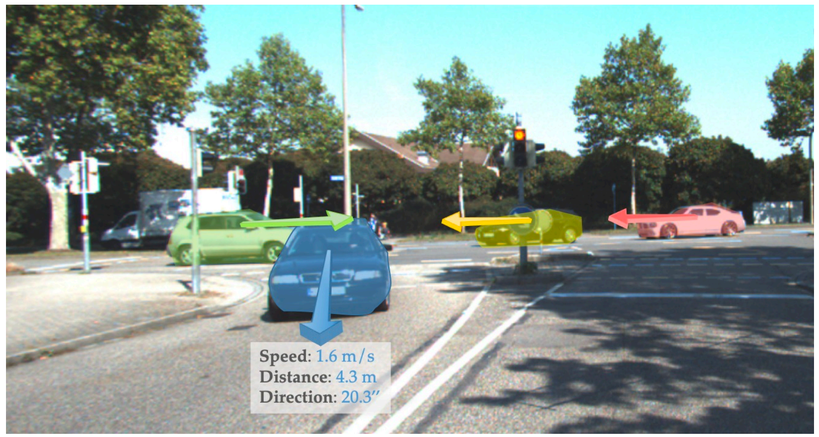
Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

3D Scene Flow: Some takeaways

Scene flow allows predicting optical flow



Typical motions are piecewise rigid



Possible parametrization as pointwise SE3 field (instead of pointwise translation)



Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

Other notable work

Tracking Everything Everywhere All at Once

Qianqian Wang[1,2] Yen-Yu Chang[1] Ruojin Cai[1] Zhengqi Li[2]
Bharath Hariharan[1] Aleksander Holynski[2,3] Noah Snavely[1,2]

[1]Cornell University [2]Google Research [3]UC Berkeley



Figure 1: We present a new method for estimating full-length motion trajectories for every pixel in every frame of a video, as illustrated by the motion paths shown above. For clarity, we only show sparse trajectories for foreground objects, though our method computes motion for *all* pixels. Our method yields accurate, coherent long-range motion even for fast-moving objects, and robustly tracks through occlusions as shown in the *dog* and *swing* examples. For context, in the second row we depict the moving object at different moments in time.

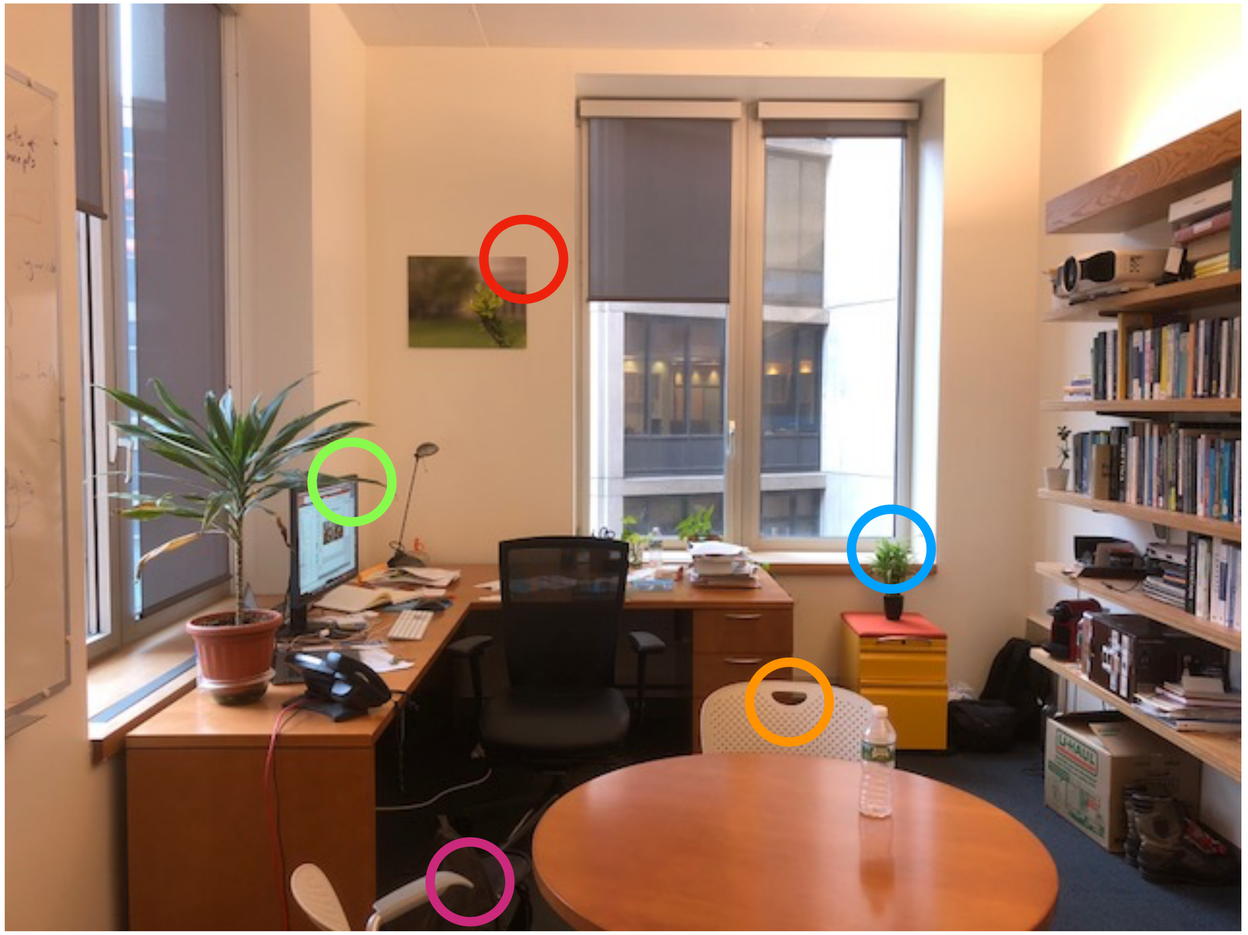
A bit of motion, a bit of 3D

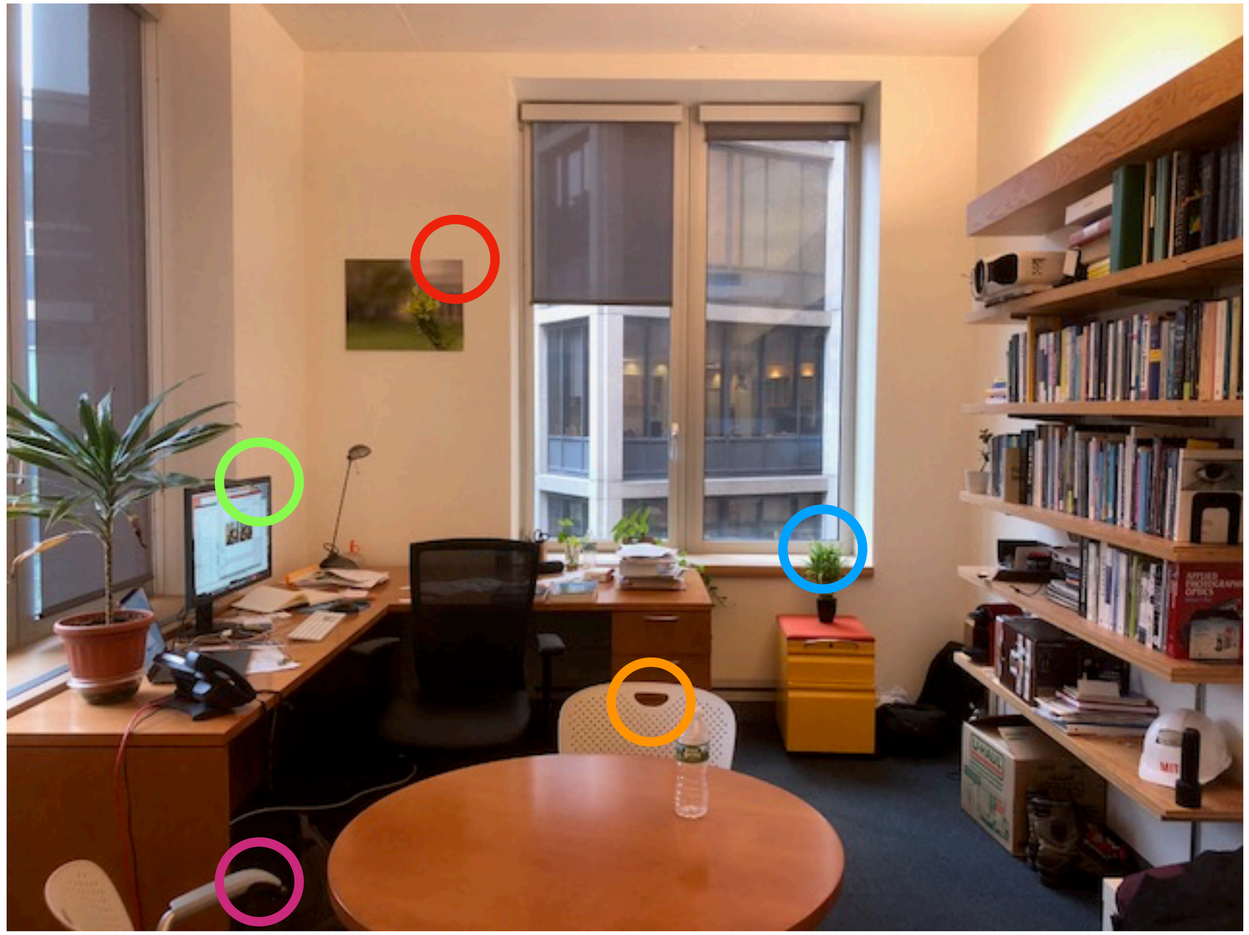
https://omnimotion.github.io/

Families of correspondence problem(s)

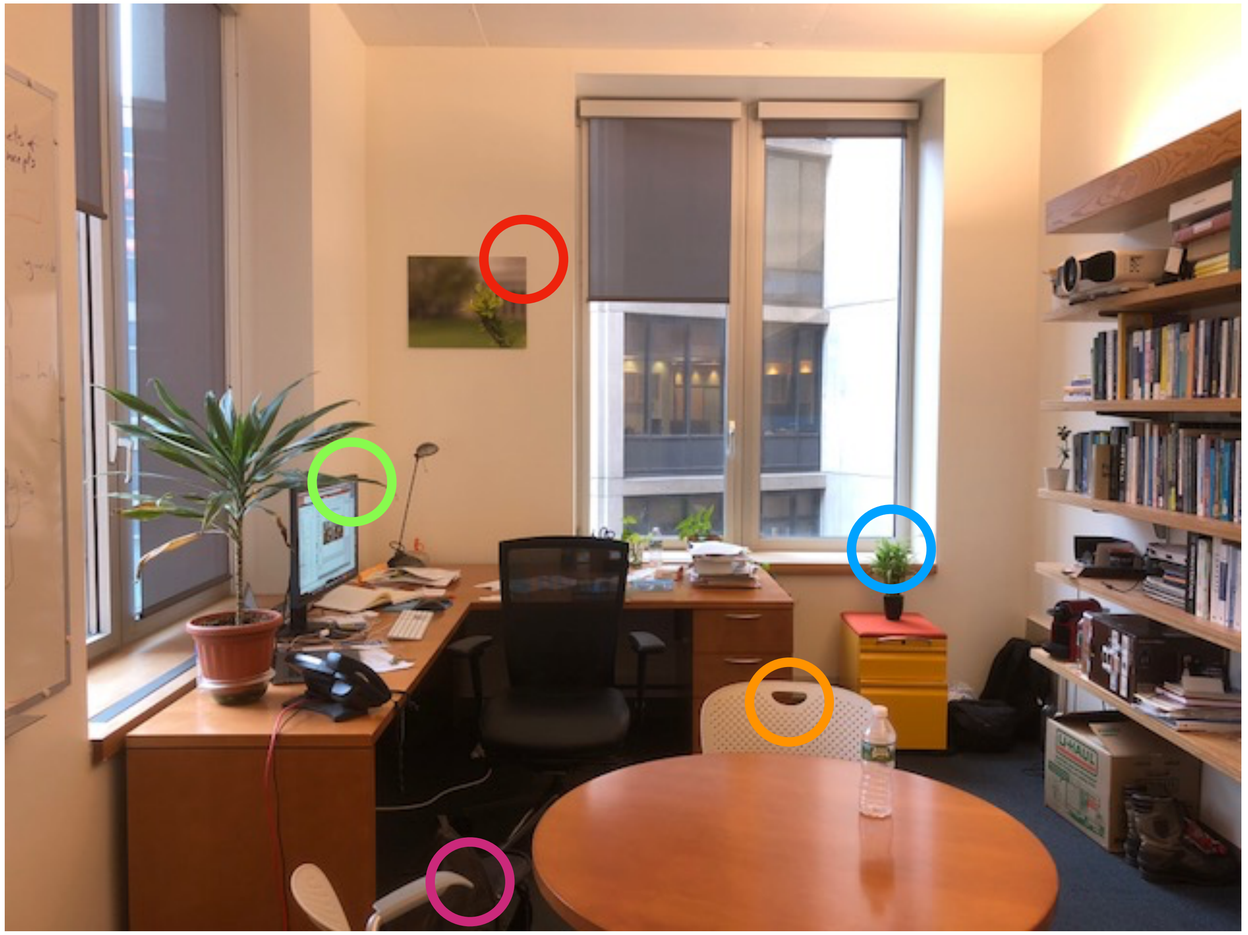
	No 3D, motion	Static 3D
Dense (, stereo)	optical flow	Stereo
Sparse and smooth (+visibility)	Motion tracking	SLAM
Sparse and far	Panorama stitching	Structure From Motion

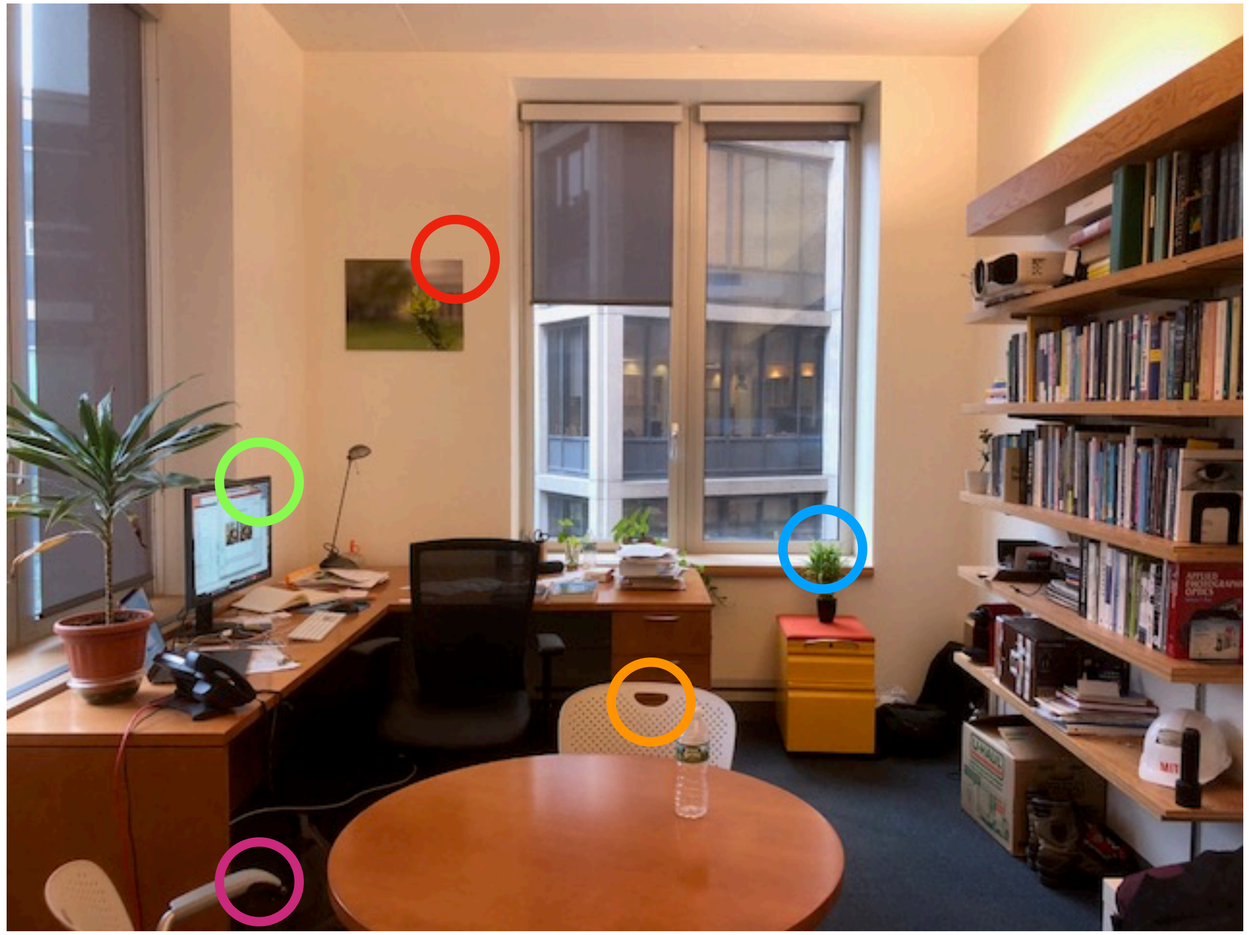
What if it’s not video, but multi-view images?





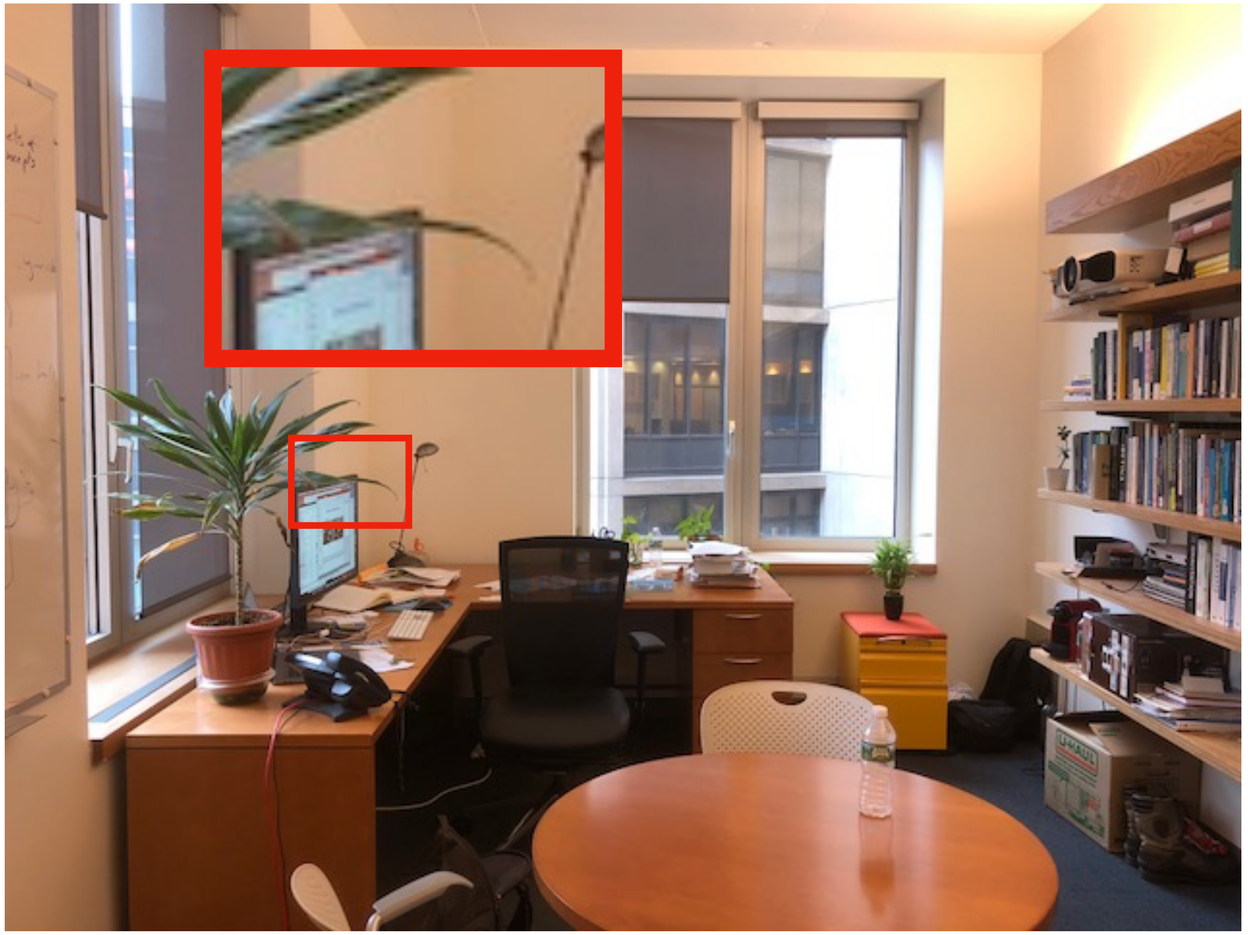
What if it’s not video, but multi-view images?





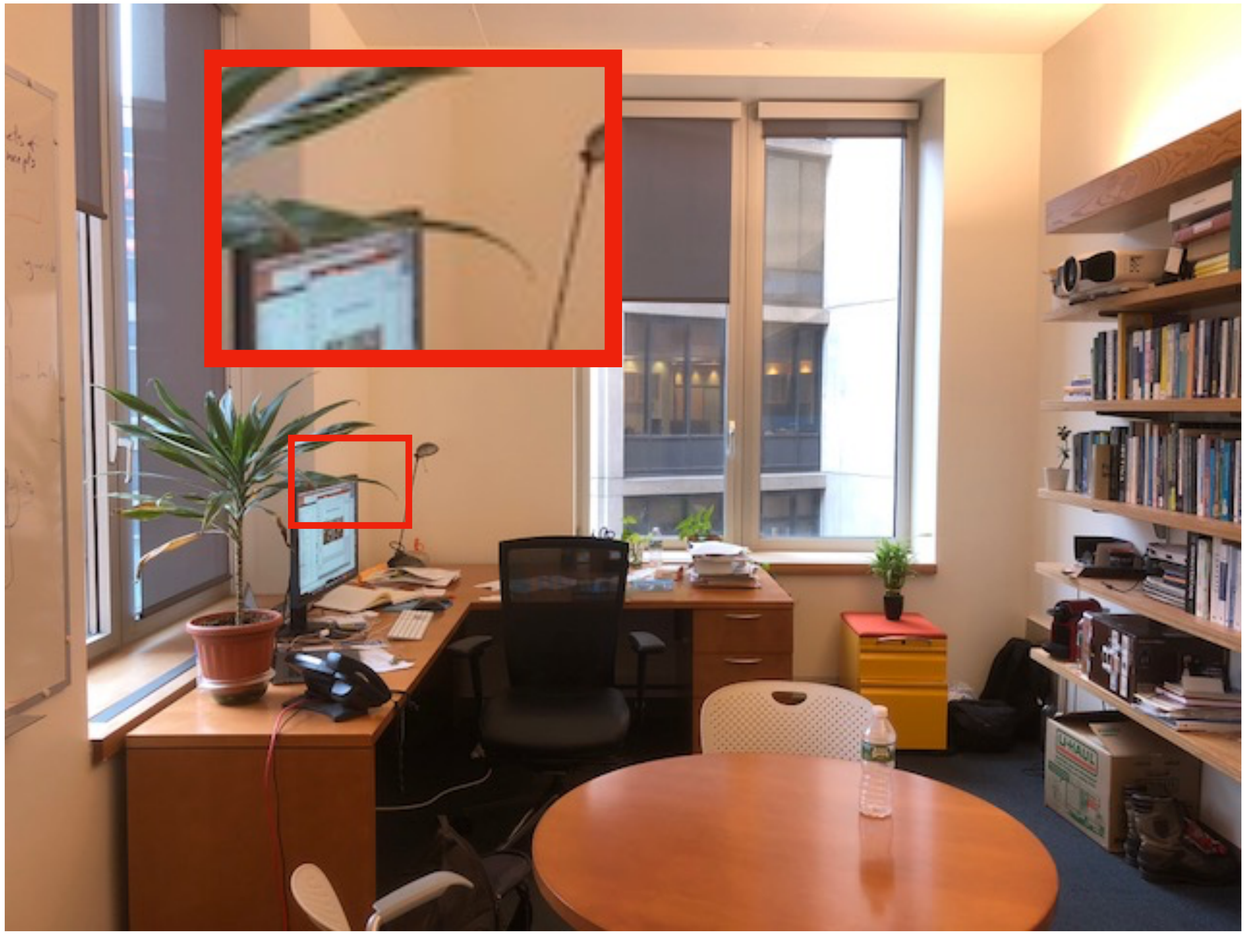
Correspondences aren’t *local* anymore. (Even less local)

What if it’s not video, but multi-view images?





What if it’s not video, but multi-view images?





Patches don’t match anymore due to camera transformations.

What if it’s not video, but multi-view images?





What if it’s not video, but multi-view images?





We want to be robust to view-dependent effects and lighting changes.

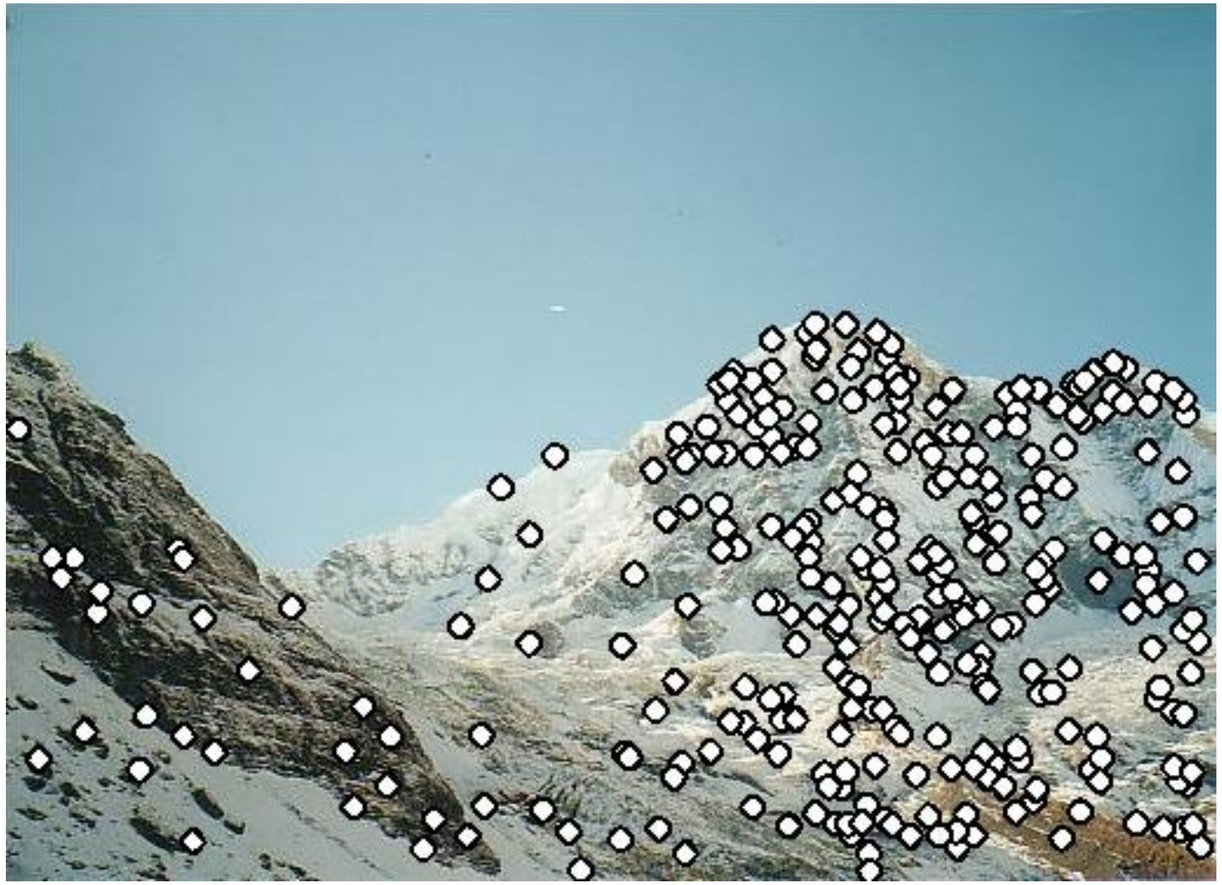
Feature Detection, Description & Matching

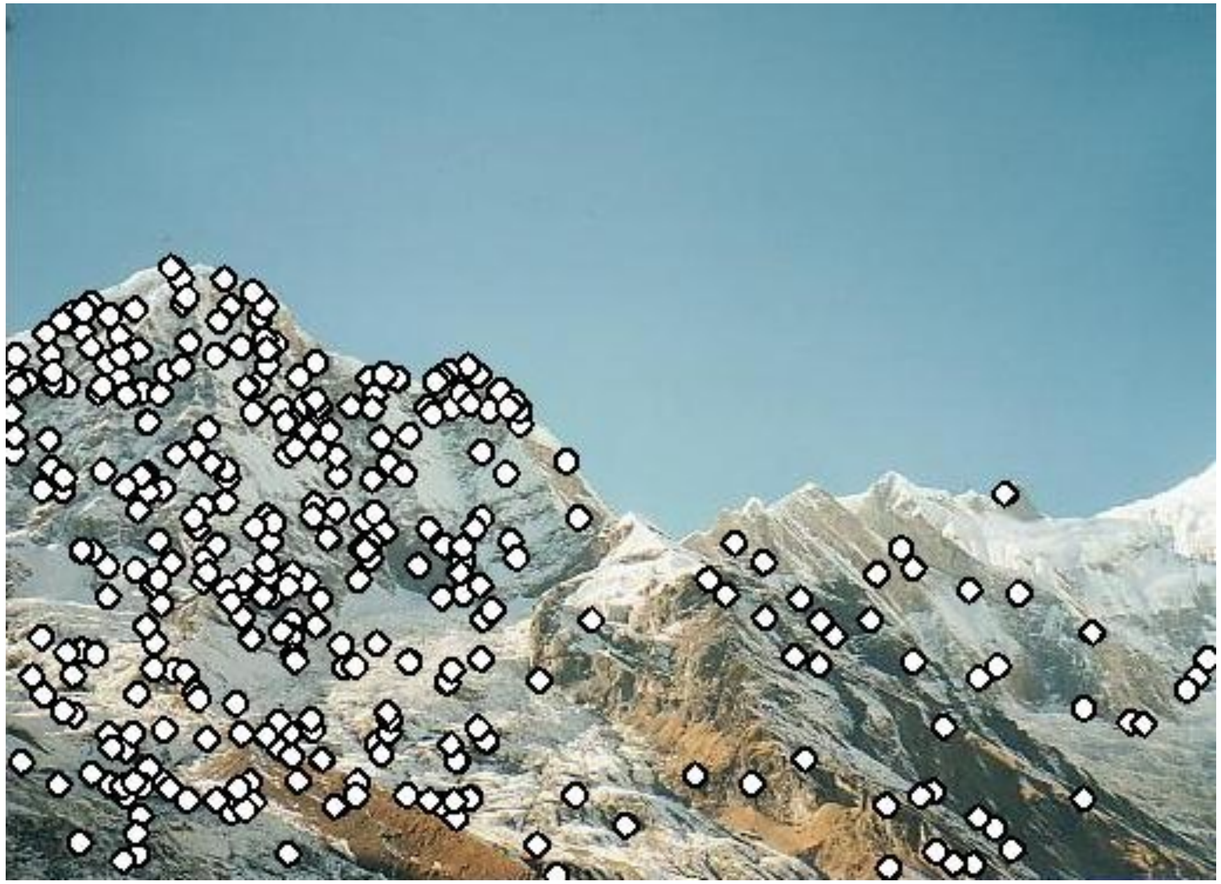




Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

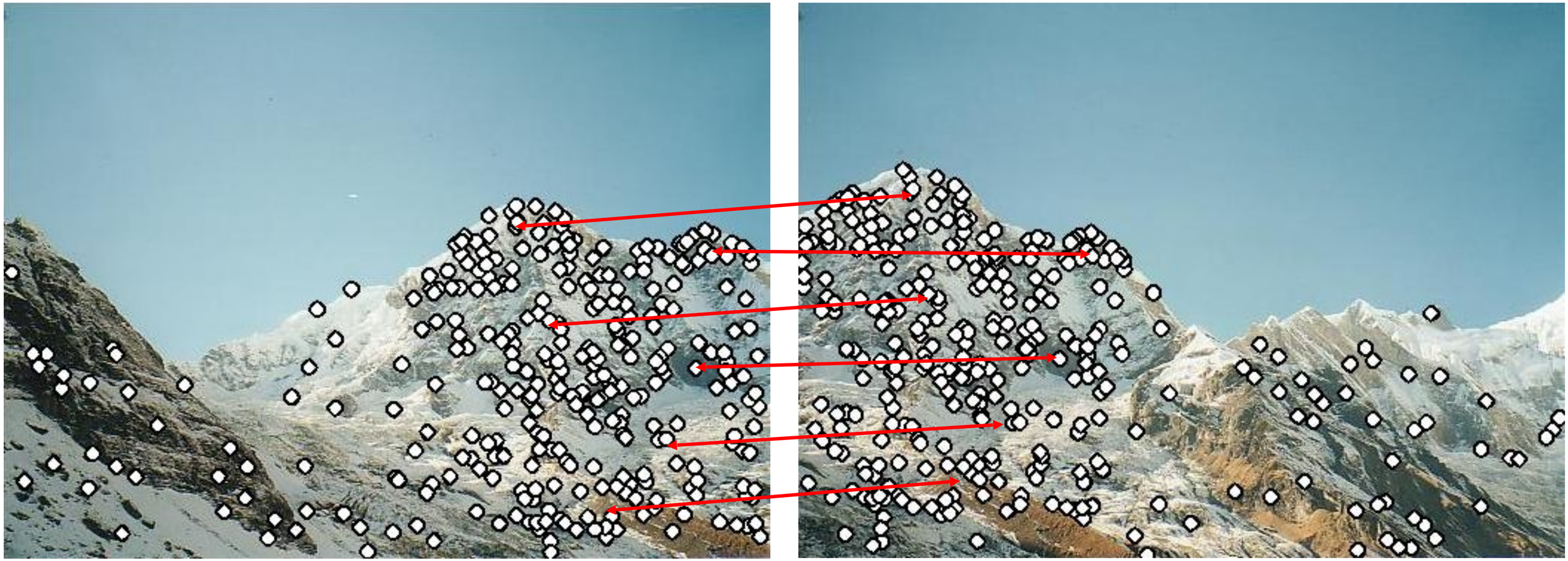
Feature Detection, Description & Matching





Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Feature Detection, Description & Matching



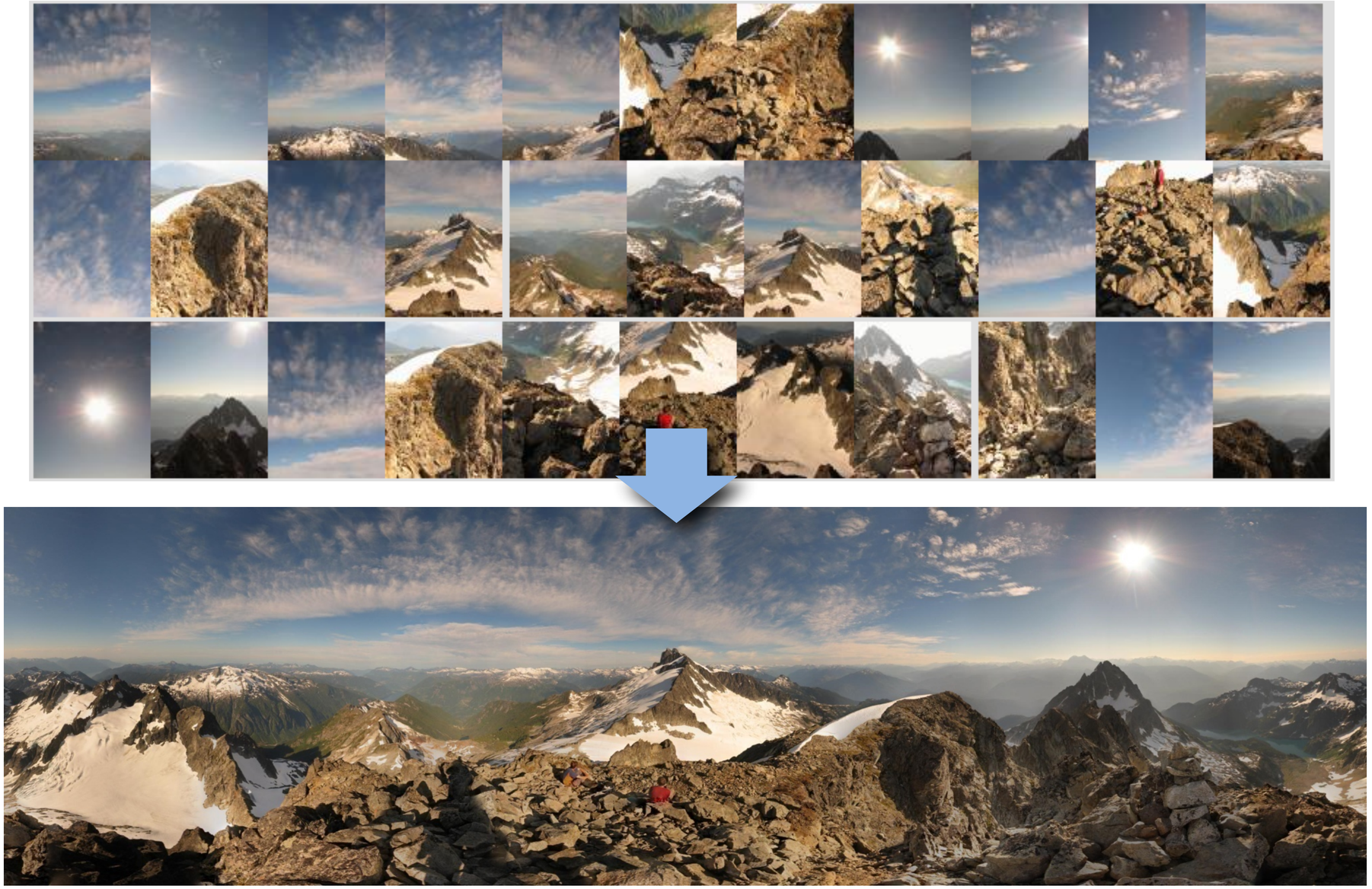
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Application 1: Panorama Stitching



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

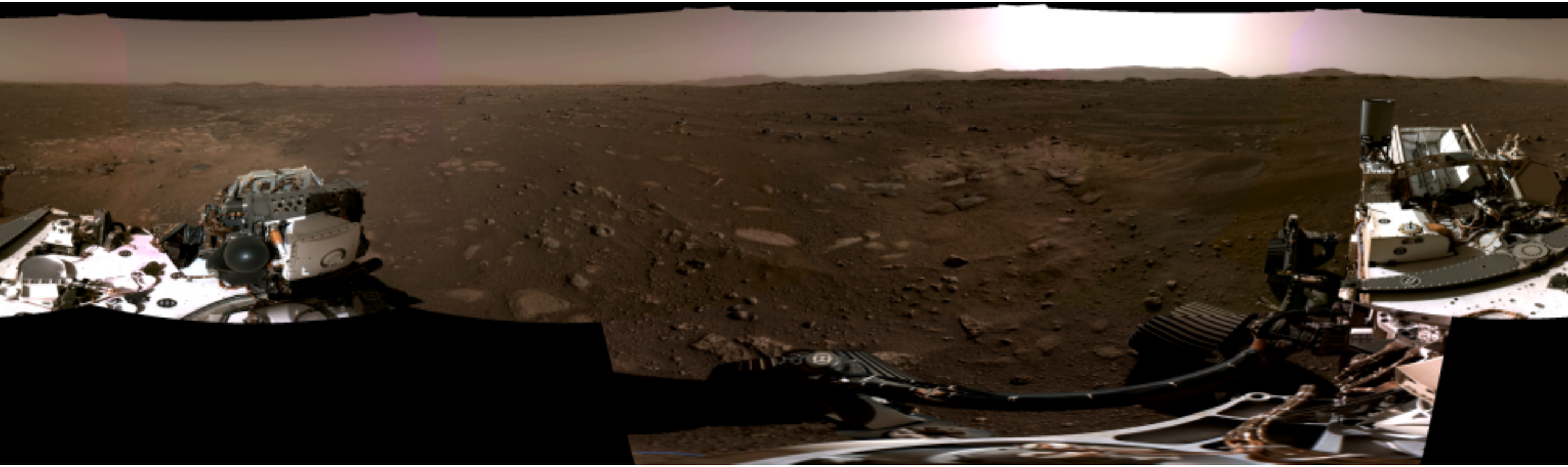
More Automatic panoramas



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Credit: Matt Brown

Panorama stitching

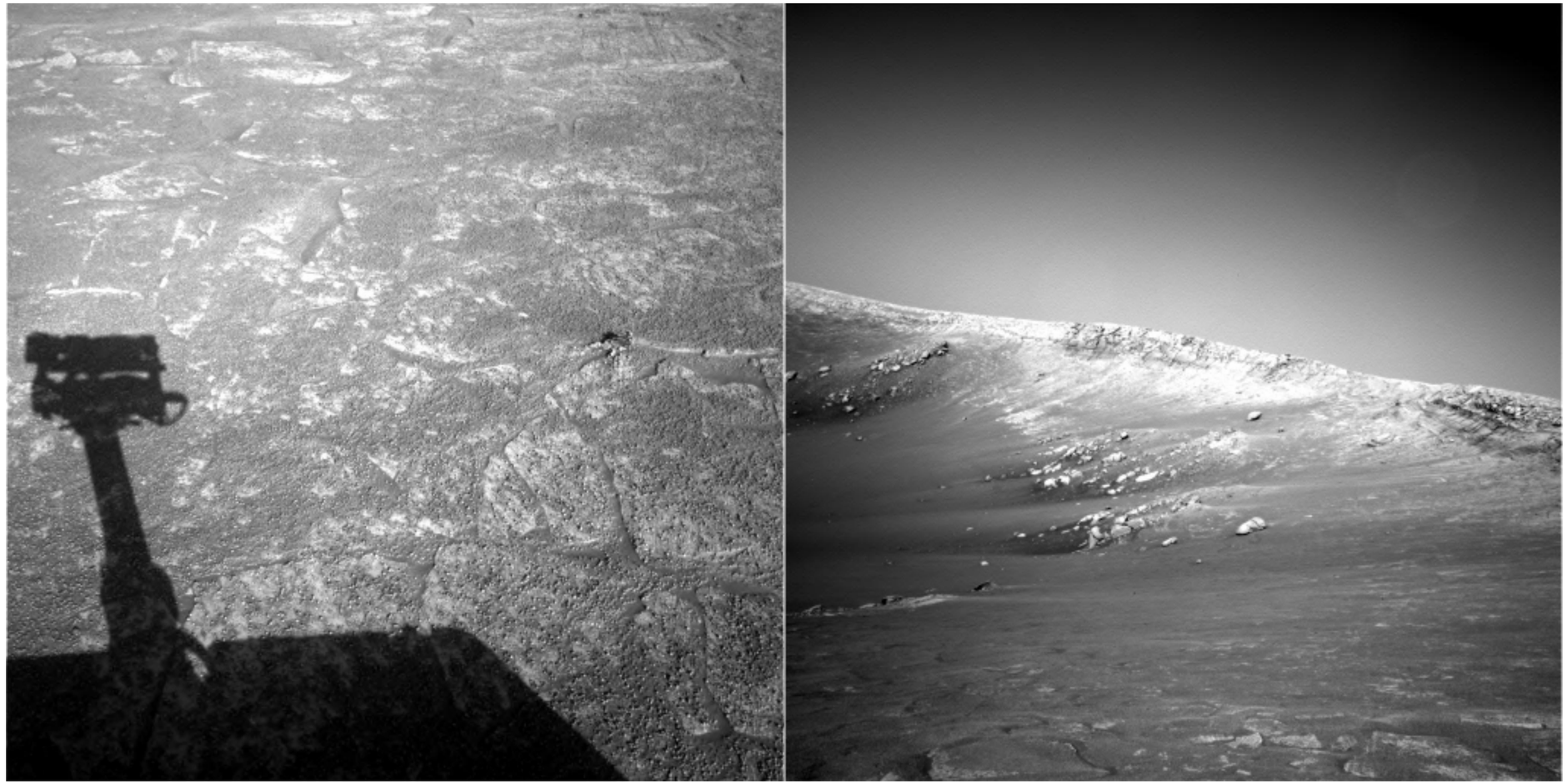


Panorama captured by Perseverence Rover, Feb. 20, 2021

https://www.space.com/nasa-perseverance-rover-first-panorama-mars

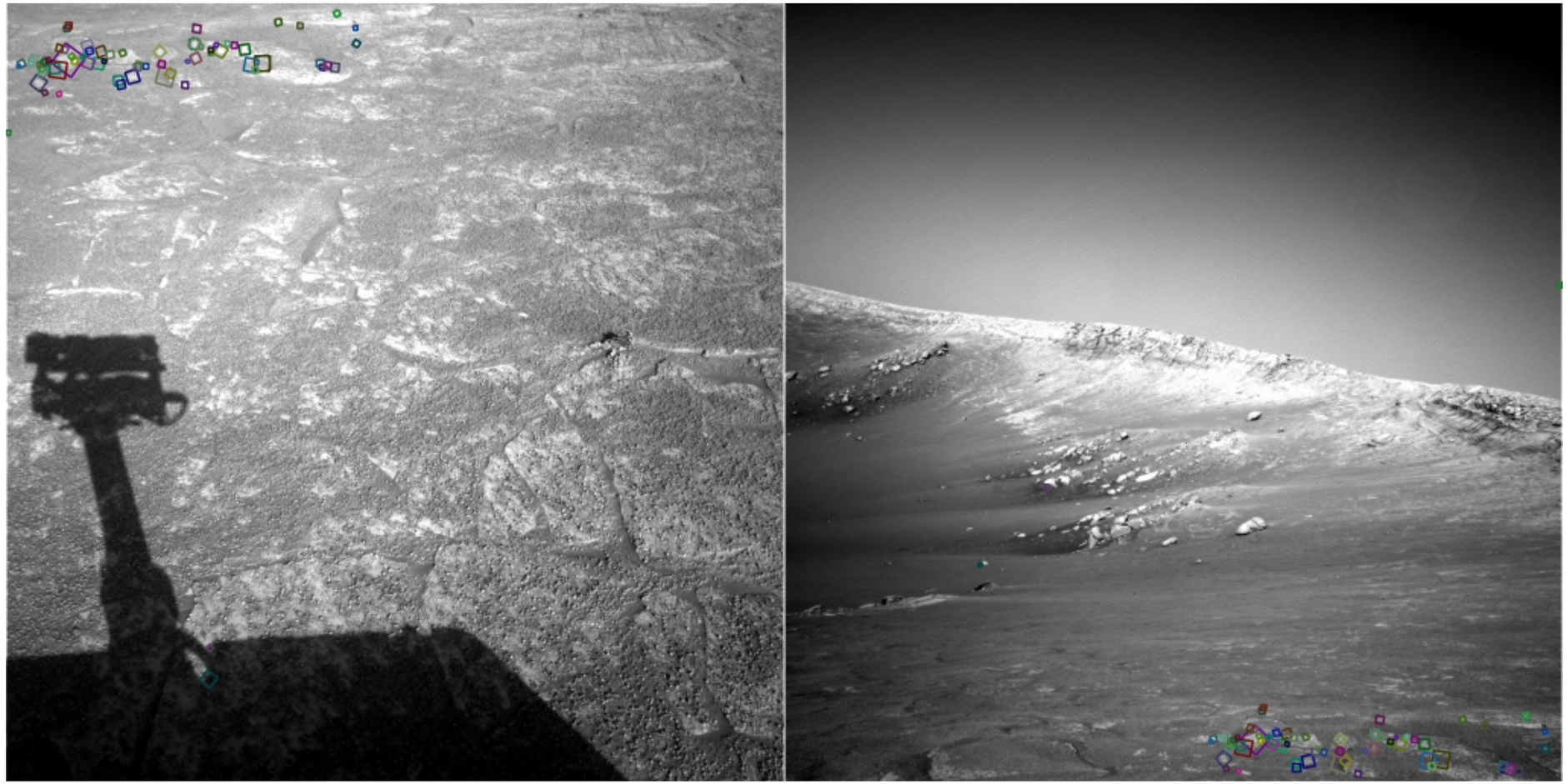
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Do these images overlap?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Answer below (look for tiny colored squares…)



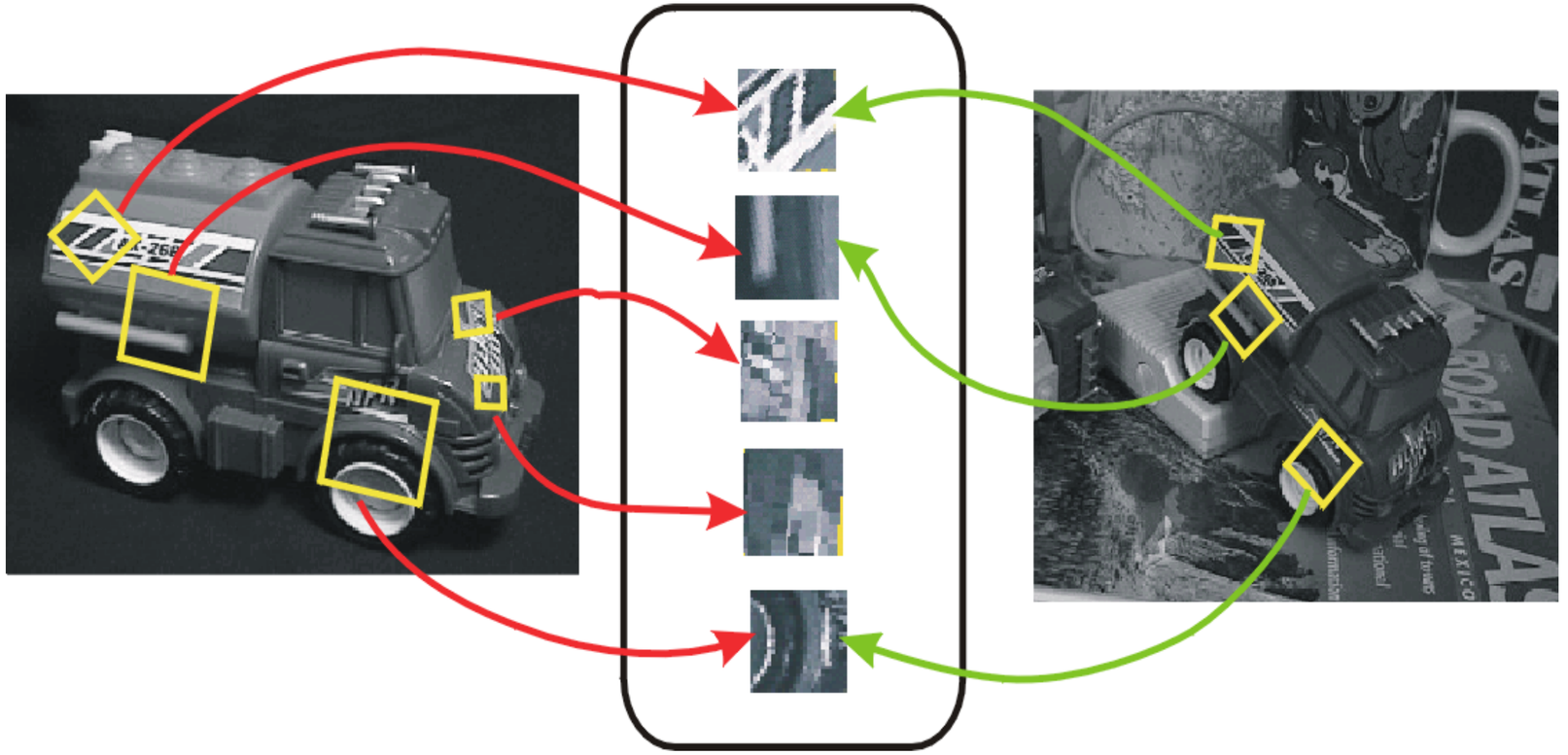
NASA Mars Rover images
with SIFT feature matches

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Invariant local features

Find features that are invariant to transformations

- geometric invariance: translation, rotation, scale
- photometric invariance: brightness, exposure, …

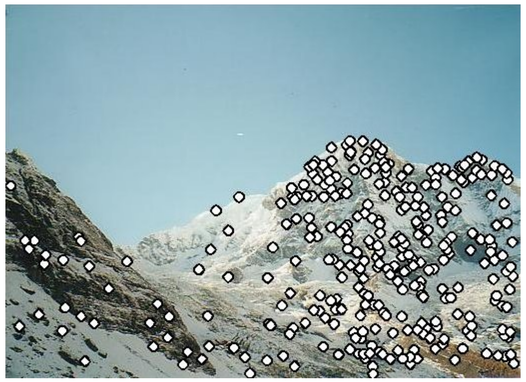


Feature Descriptors

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Local features: main components

1) **Detection**: Identify the interest points

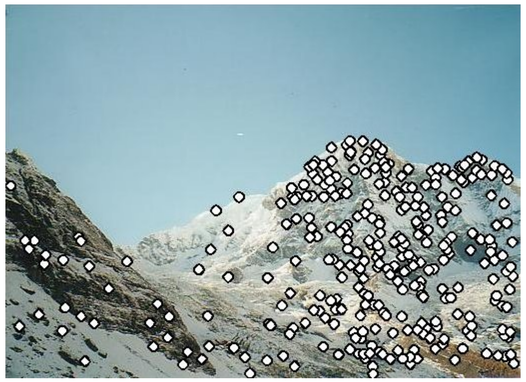


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

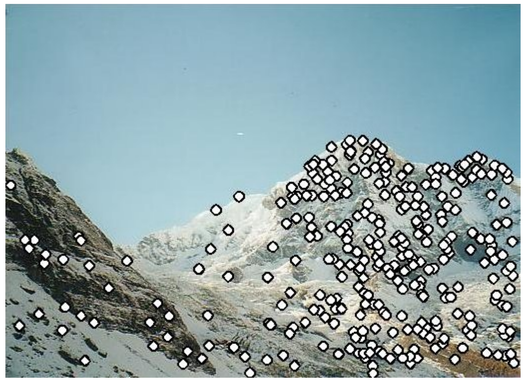
Credit: Kristen Grauman

Local features: main components

1) **Detection**: Identify the interest points



2) **Description**: Extract vector feature descriptor surrounding each interest point

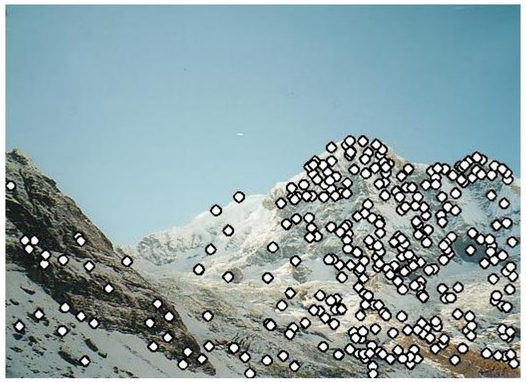


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

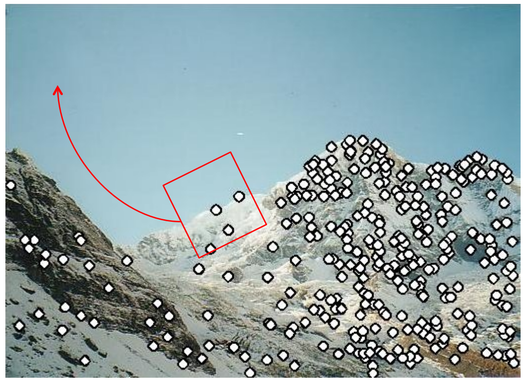
Credit: Kristen Grauman

Local features: main components

1) **Detection**: Identify the interest points



2) **Description**: Extract vector feature descriptor surrounding each interest point

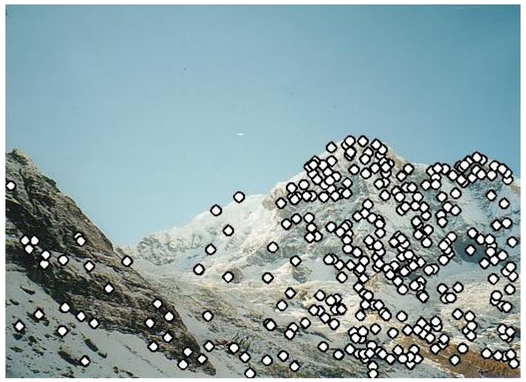


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

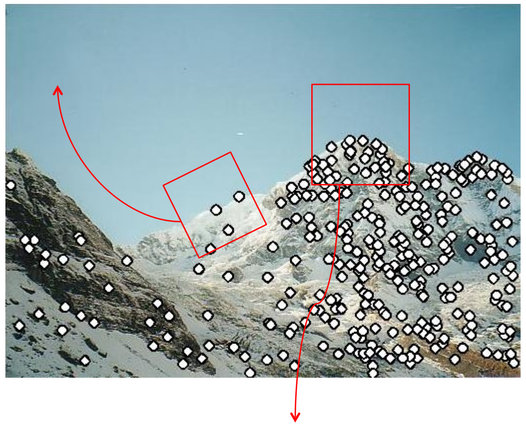
Credit: Kristen Grauman

Local features: main components

1) **Detection**: Identify the interest points



2) **Description**: Extract vector feature descriptor surrounding each interest point

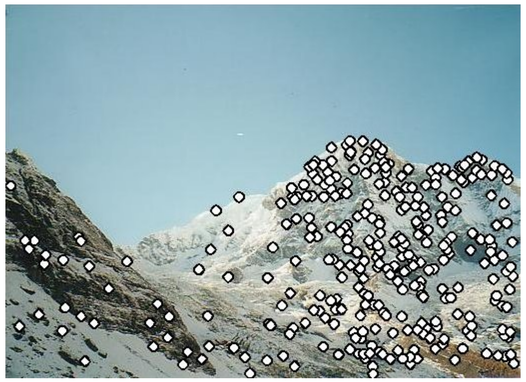


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

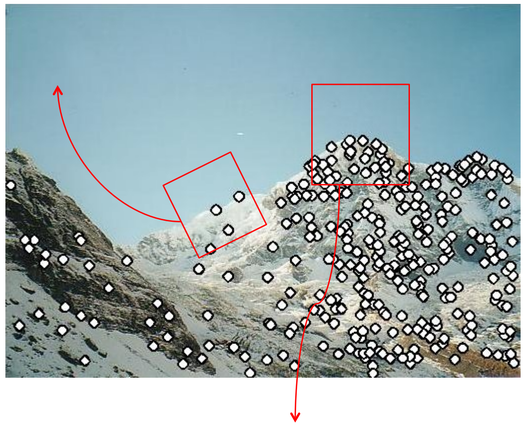
Credit: Kristen Grauman

Local features: main components

1) **Detection**: Identify the interest points



2) **Description**: Extract vector feature descriptor surrounding each interest point



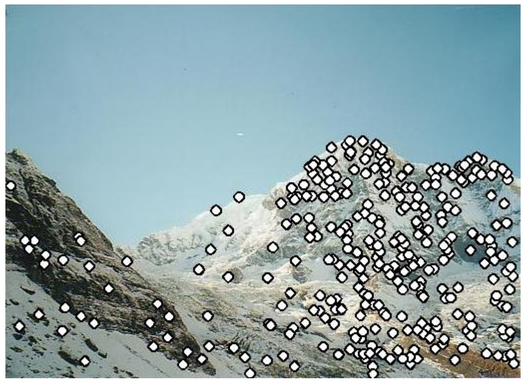
3) **Matching**: Determine correspondence between descriptors in two views

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

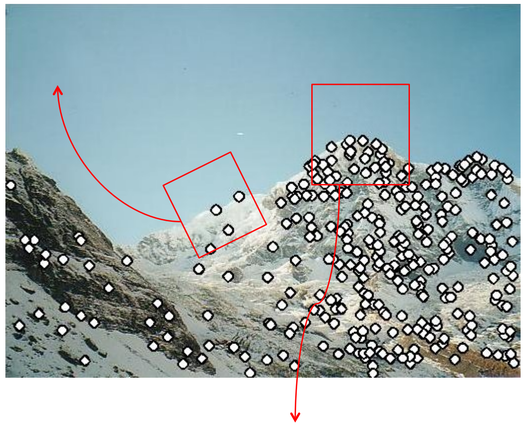
Credit: Kristen Grauman

Local features: main components

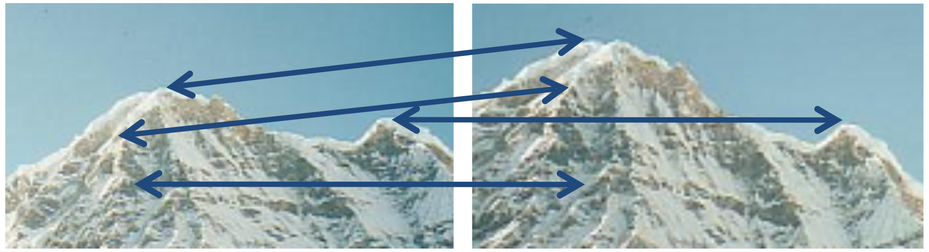
1) **Detection**: Identify the interest points



2) **Description**: Extract vector feature descriptor surrounding each interest point



3) **Matching**: Determine correspondence between descriptors in two views

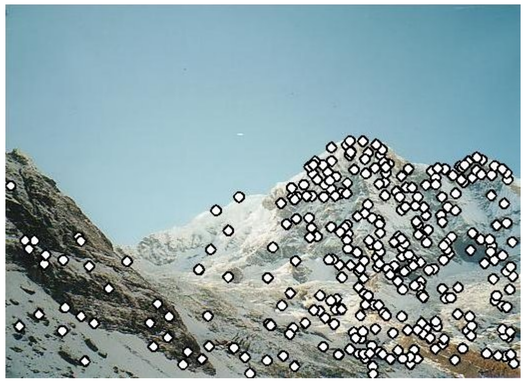


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Credit: Kristen Grauman

Local features: main components

1) **Detection: Identify the interest points**

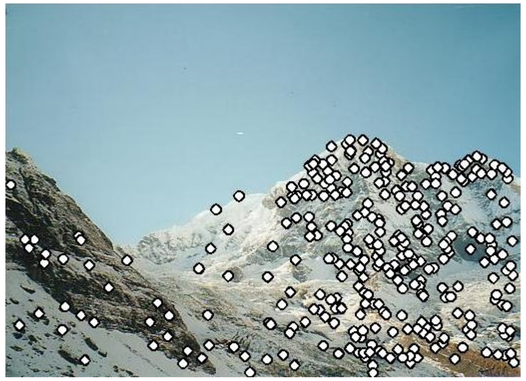


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

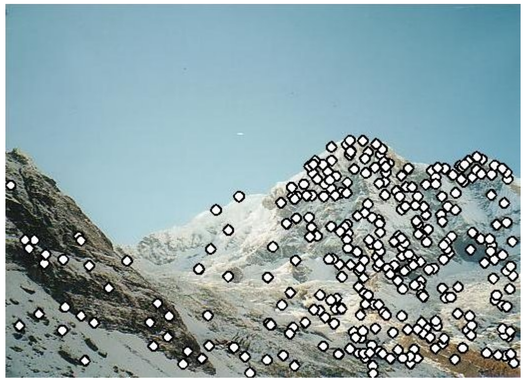
Credit: Kristen Grauman

Local features: main components

1) **Detection: Identify the interest points**



2) **Description**: Extract vector feature descriptor surrounding each interest point

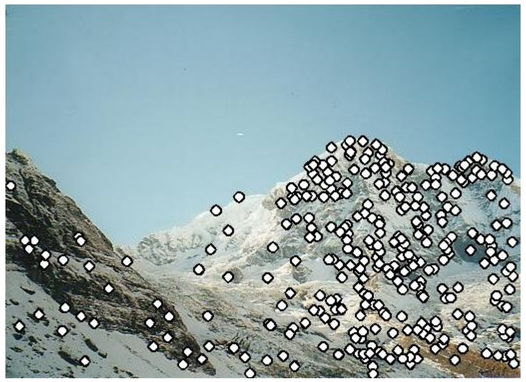


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

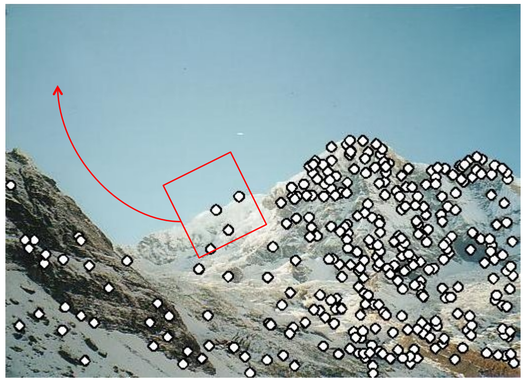
Credit: Kristen Grauman

Local features: main components

1) **Detection: Identify the interest points**



2) **Description**: Extract vector feature descriptor surrounding each interest point

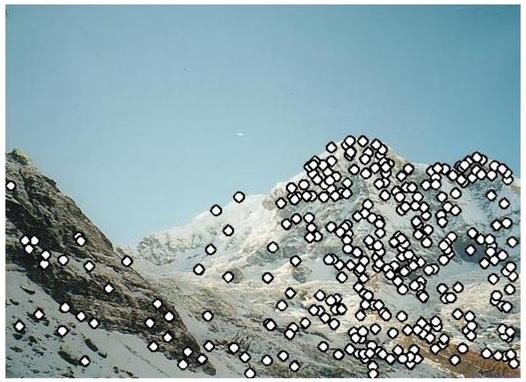


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

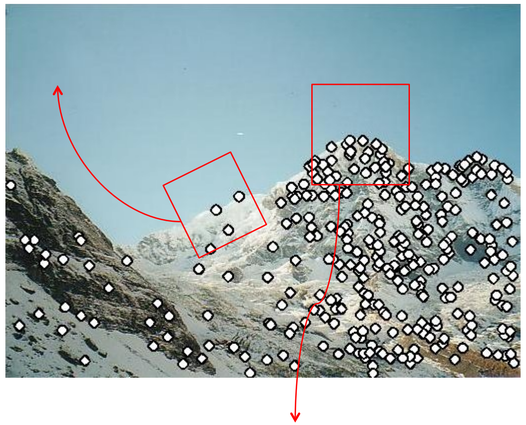
Credit: Kristen Grauman

Local features: main components

1) **Detection: Identify the interest points**



2) **Description**: Extract vector feature descriptor surrounding each interest point

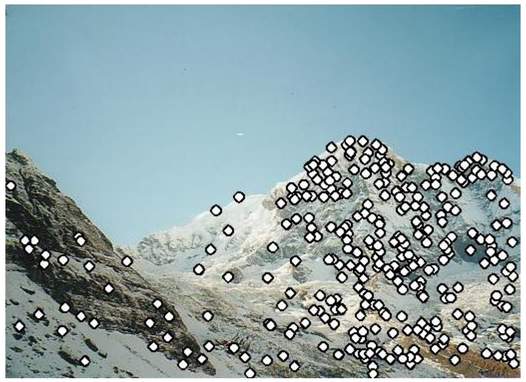


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

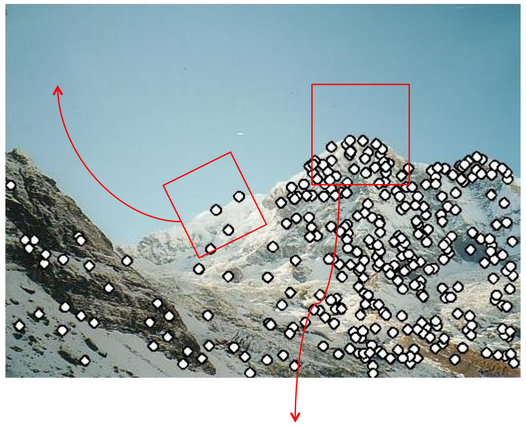
Credit: Kristen Grauman

Local features: main components

1) **Detection: Identify the interest points**



2) **Description**: Extract vector feature descriptor surrounding each interest point



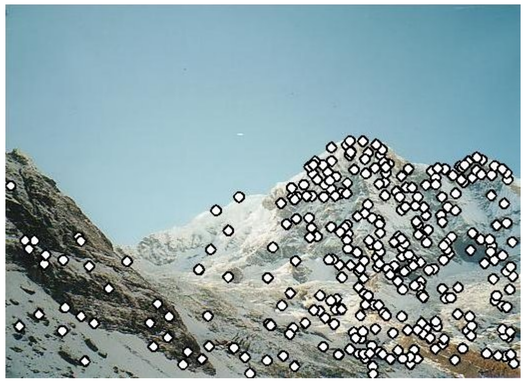
3) **Matching**: Determine correspondence between descriptors in two views

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

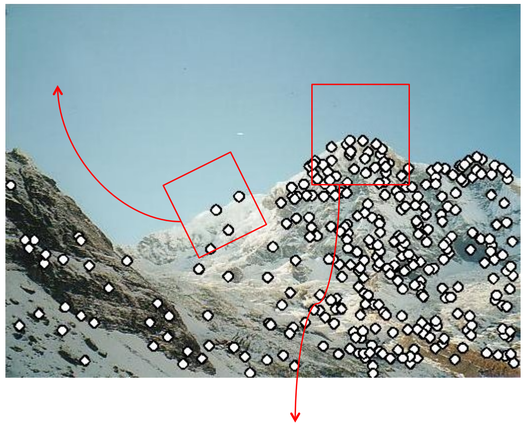
Credit: Kristen Grauman

Local features: main components

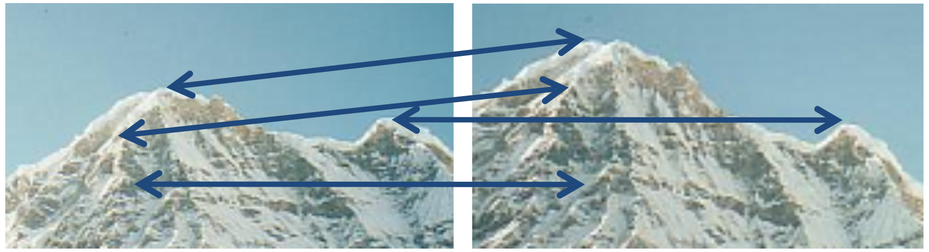
1) **Detection: Identify the interest points**



2) **Description**: Extract vector feature descriptor surrounding each interest point



3) **Matching**: Determine correspondence between descriptors in two views



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Credit: Kristen Grauman

What makes a good feature?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Want uniqueness

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Want uniqueness

Look for image regions that are unusual

– Lead to unambiguous matches in other images

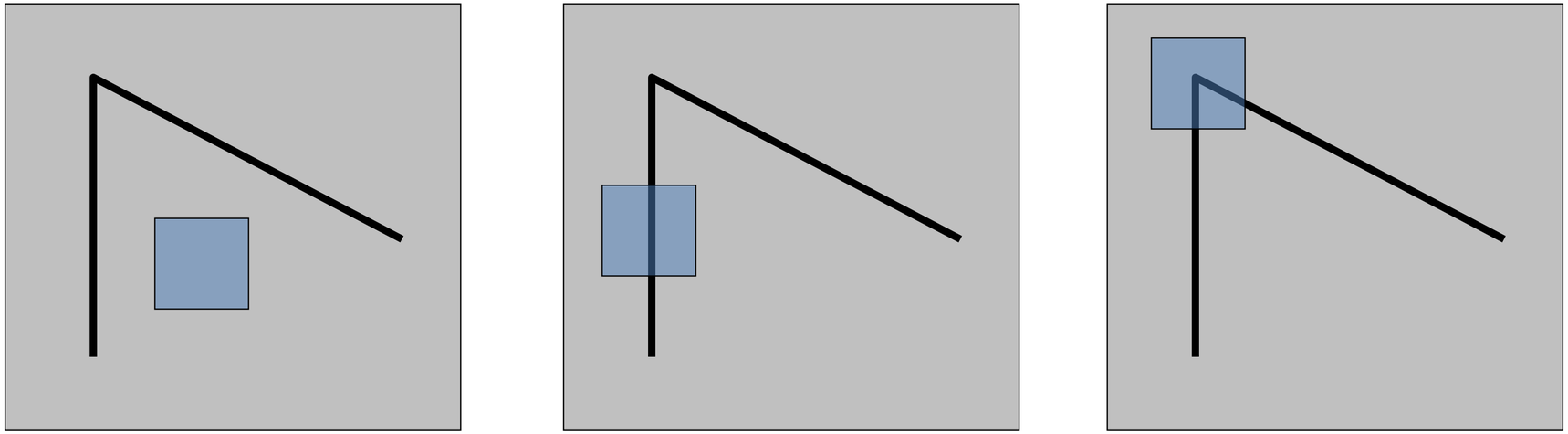
How to define “unusual”?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Local measures of uniqueness

Suppose we only consider a small window of pixels

– What defines whether a feature is a good or bad candidate?

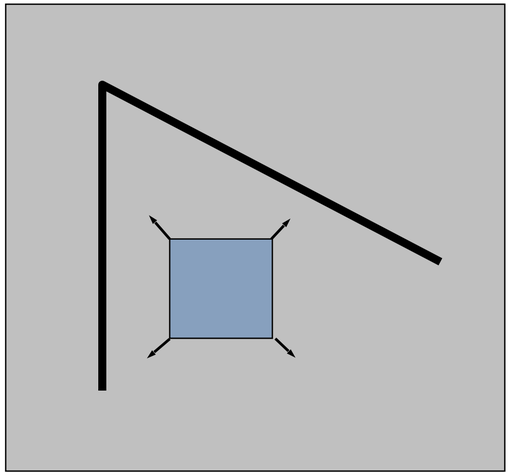


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

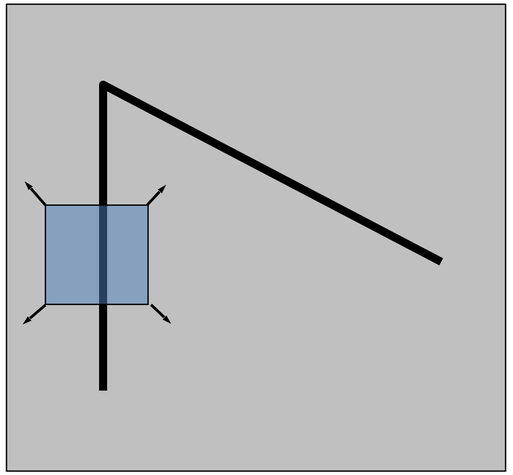
Credit: S. Seitz, D. Frolova, D. Simakov

Local measures of uniqueness

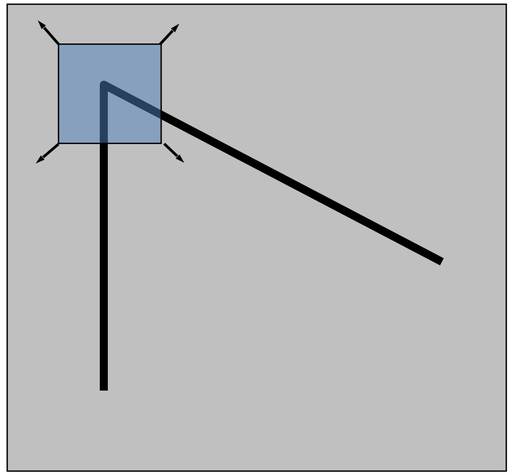
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

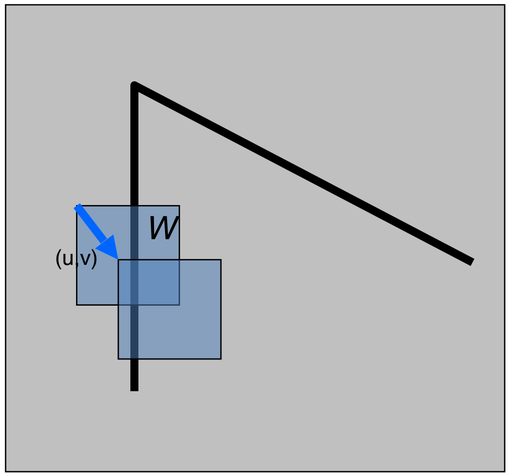
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Credit: S. Seitz, D. Frolova, D. Simakov

Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

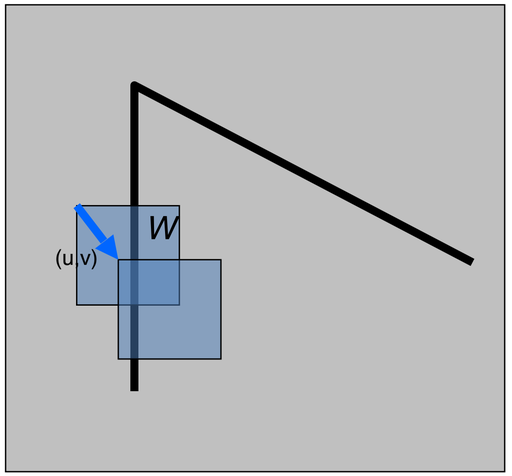
[Taylor expansion]

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

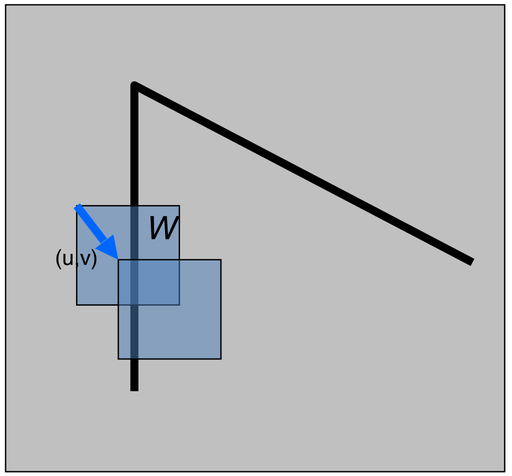
$$\text{[Taylor expansion]} \approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:



$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

[Taylor expansion] $$\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2$$

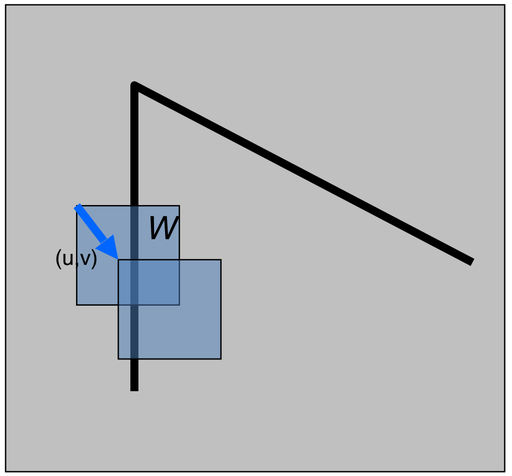
$$\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:

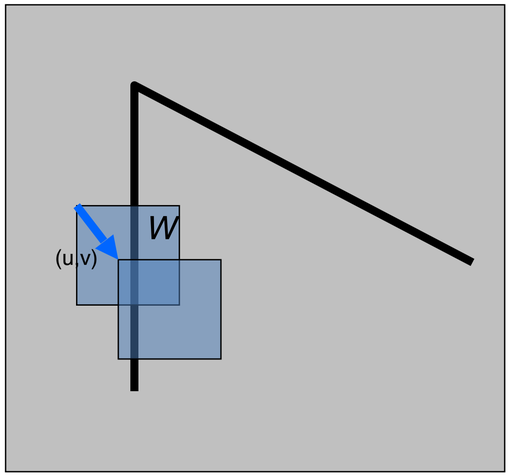


$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math



Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:

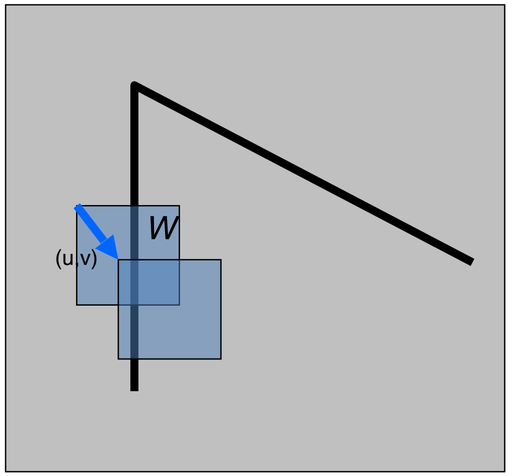
$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math



Consider shifting the window *W* by (*u*,*v*)

- define an SSD "error" *E(u,v)*:

$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

- Thus, *E*(*u*,*v*) is locally approximated as a quadratic error function

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic for

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic for

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic for

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

The second moment matrix

The surface $E(u,v)$ is locally approximated by a quadratic for

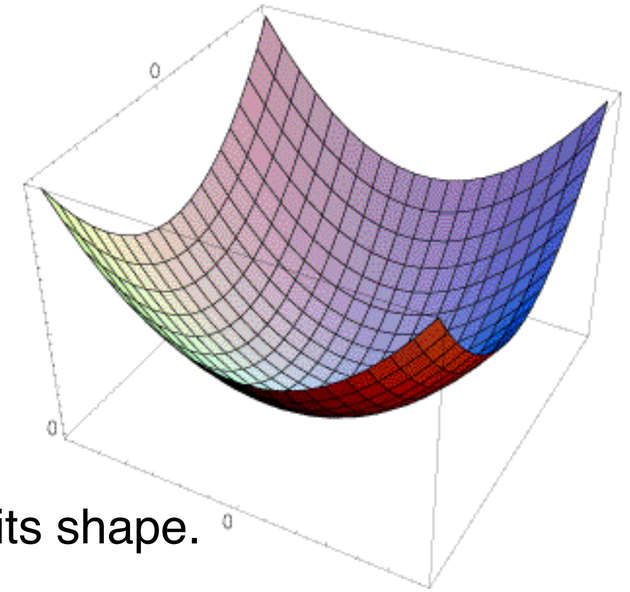
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Let's try to understand its shape.

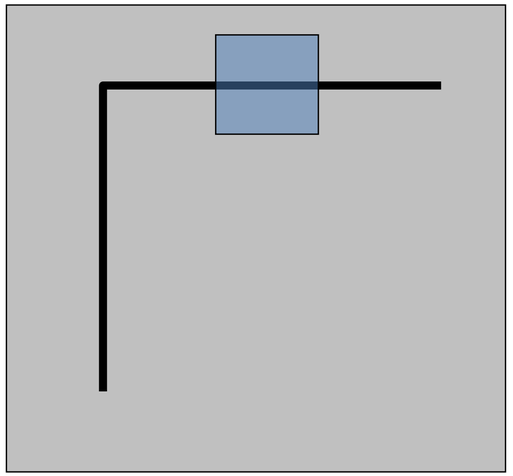
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

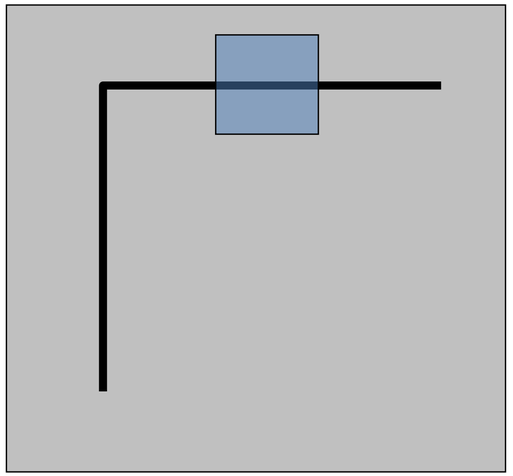
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$

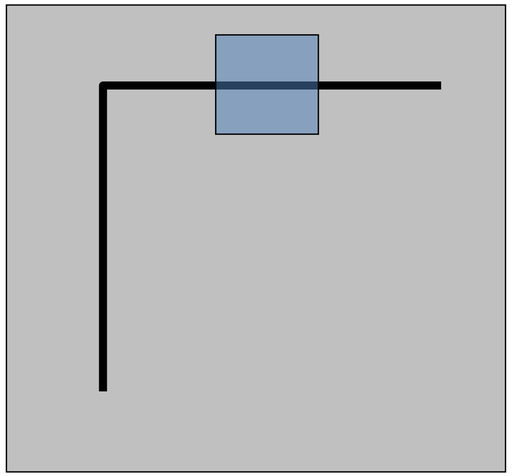
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

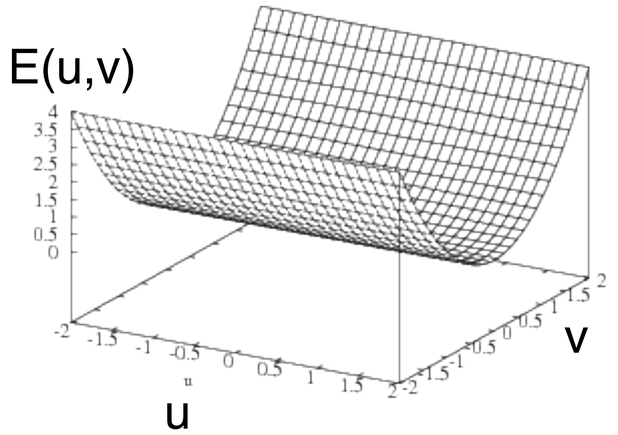
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$



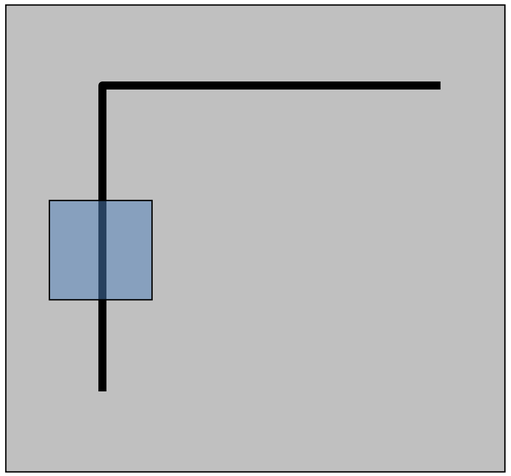
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

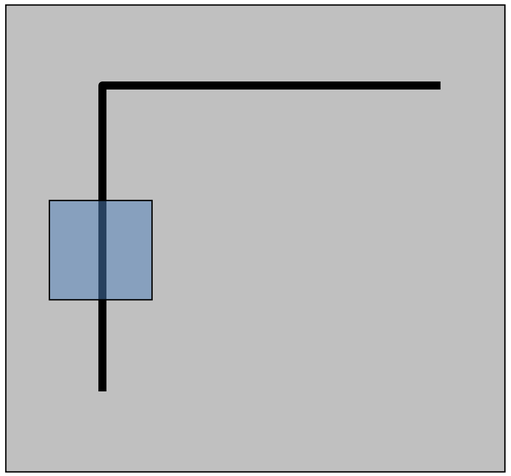
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

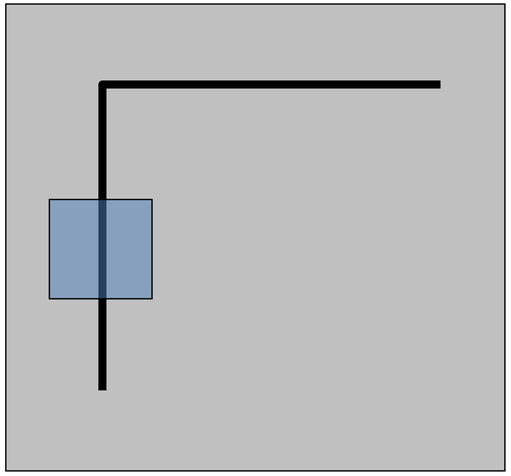
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

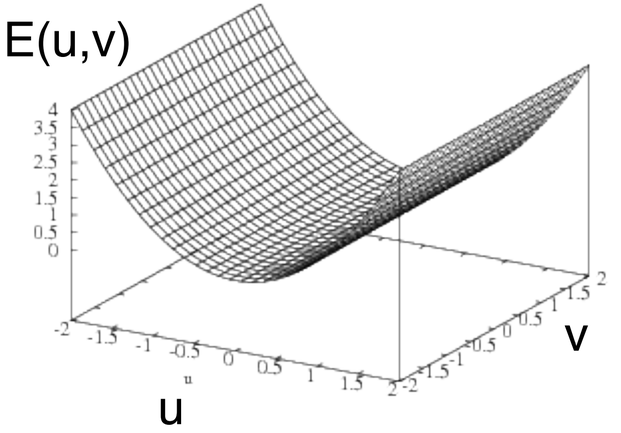
$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$

E(u,v)

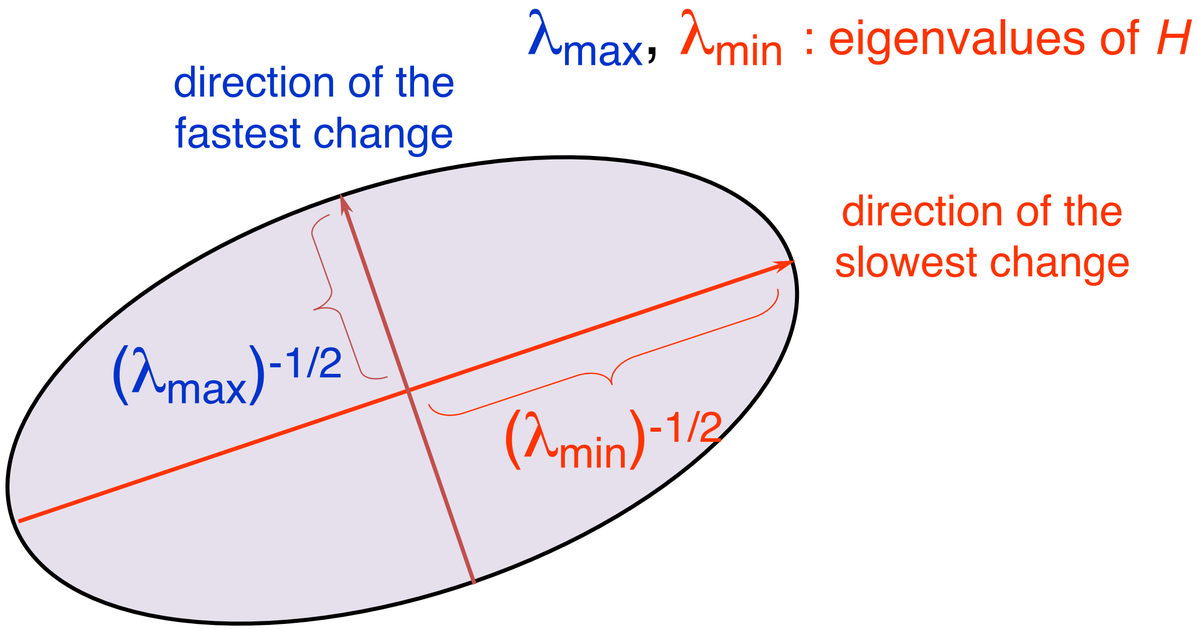


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

General case

We can visualize H as an ellipse with axis lengths determined by the *eigenvalues* of H and orientation determined by the *eigenvectors* of H

Ellipse equation:



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

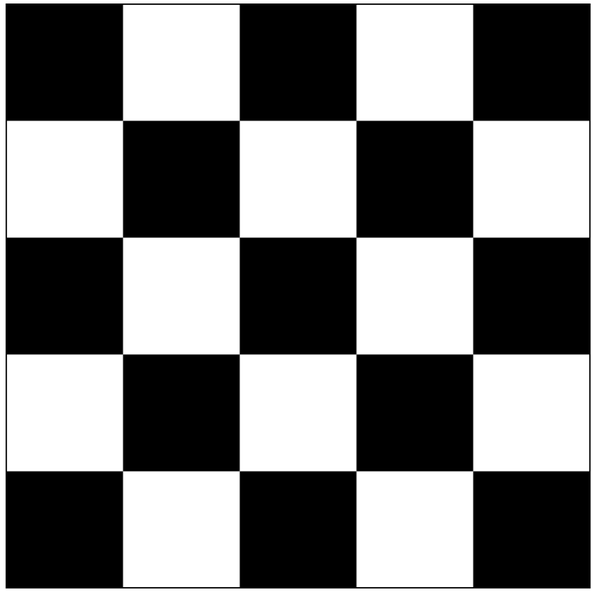
Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

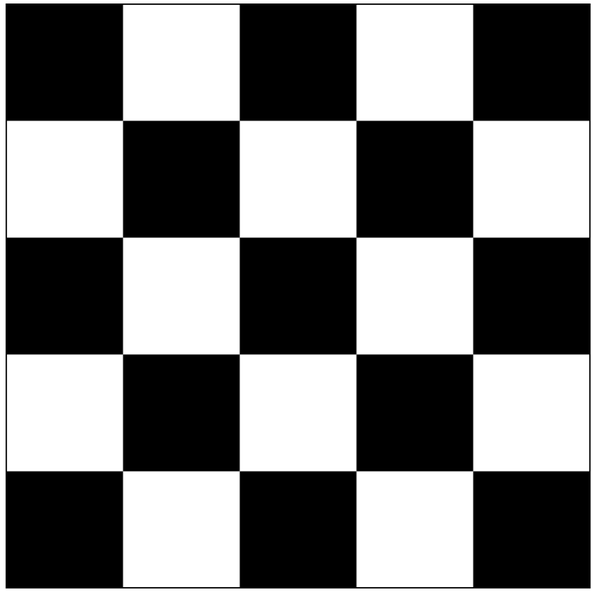
Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

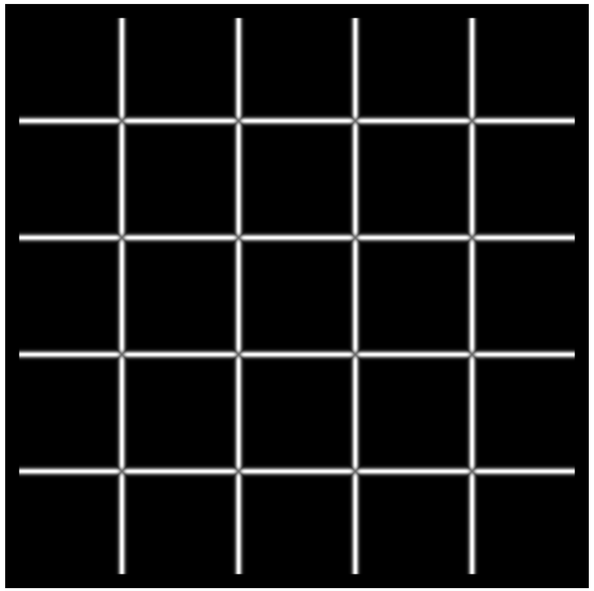
- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



I



$\lambda_{\max}$

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

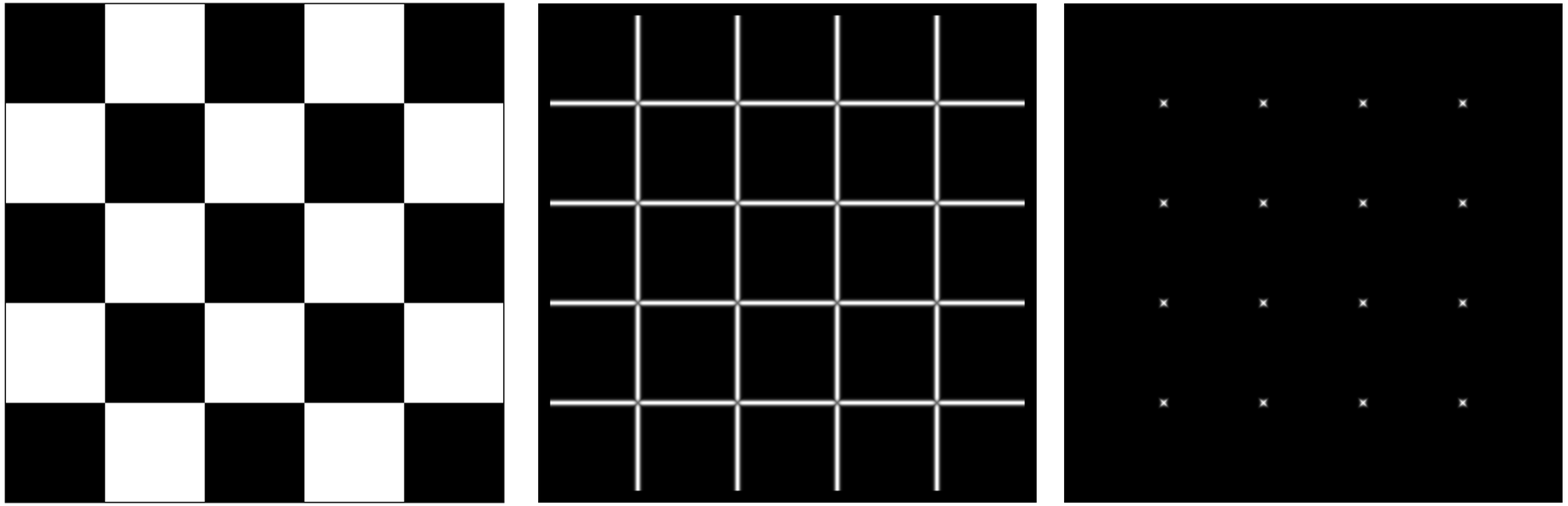
Corner detection: the math

How are λ_{max}, x_{max}, λ_{min}, and x_{min} relevant for feature detection?

- What's our feature scoring function?

Want $E(u,v)$ to be large for small shifts in all directions

- the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_{min}) of H



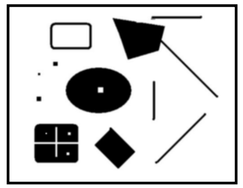
I λ_{max} λ_{min}

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Harris Detector – Overview [Harris88]

Second moment matrix

0. Input image

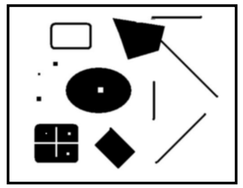


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

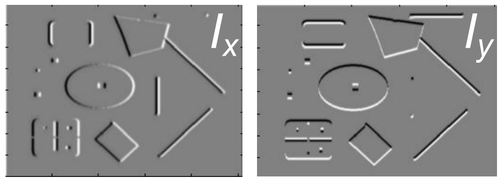
Harris Detector – Overview [Harris88]

Second moment matrix

0. Input image



1. Image derivatives

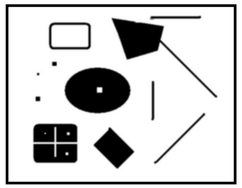


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

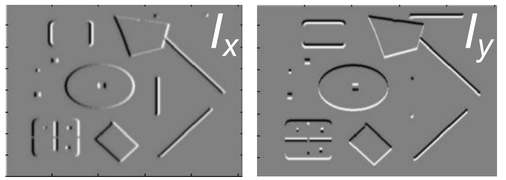
Harris Detector – Overview [Harris88]

Second moment matrix

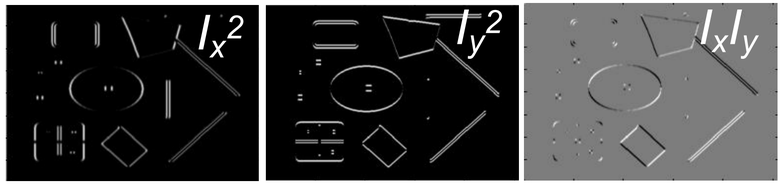
0. Input image



1. Image derivatives



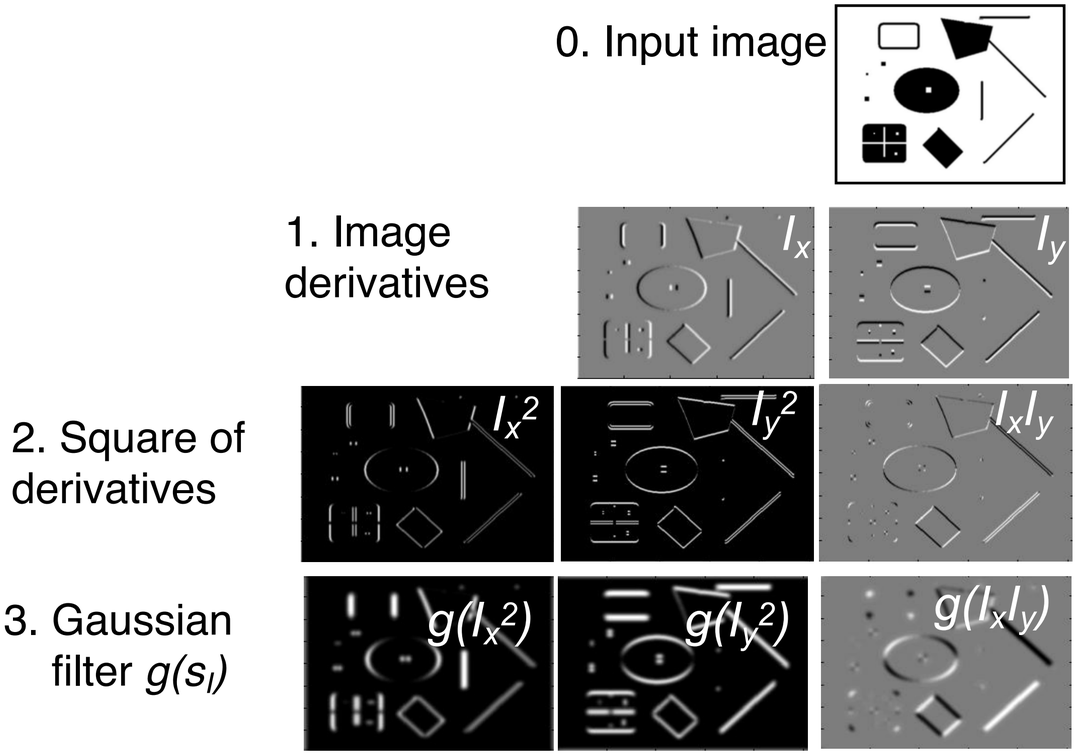
2. Square of derivatives



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Harris Detector – Overview [Harris88]

Second moment matrix

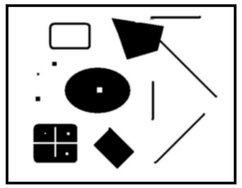


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

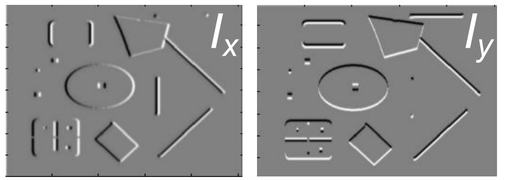
Harris Detector – Overview [Harris88]

Second moment matrix

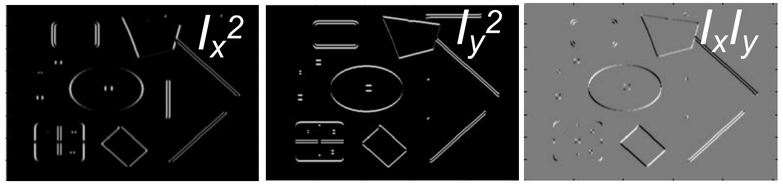
0. Input image



1. Image derivatives



2. Square of derivatives



3. Gaussian filter $g(s_I)$



4. Cornerness function – both eigenvalues are strong

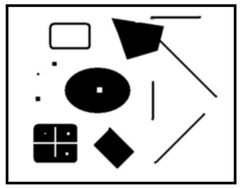


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

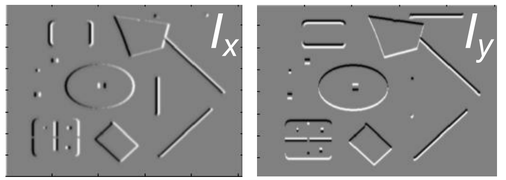
Harris Detector – Overview [Harris88]

Second moment matrix

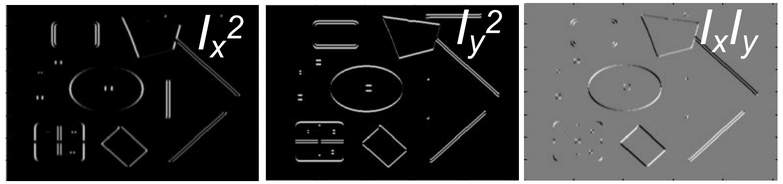
0. Input image



1. Image derivatives



2. Square of derivatives



3. Gaussian filter $g(s_I)$



4. Cornerness function – both eigenvalues are strong



5. Non-maxima suppression

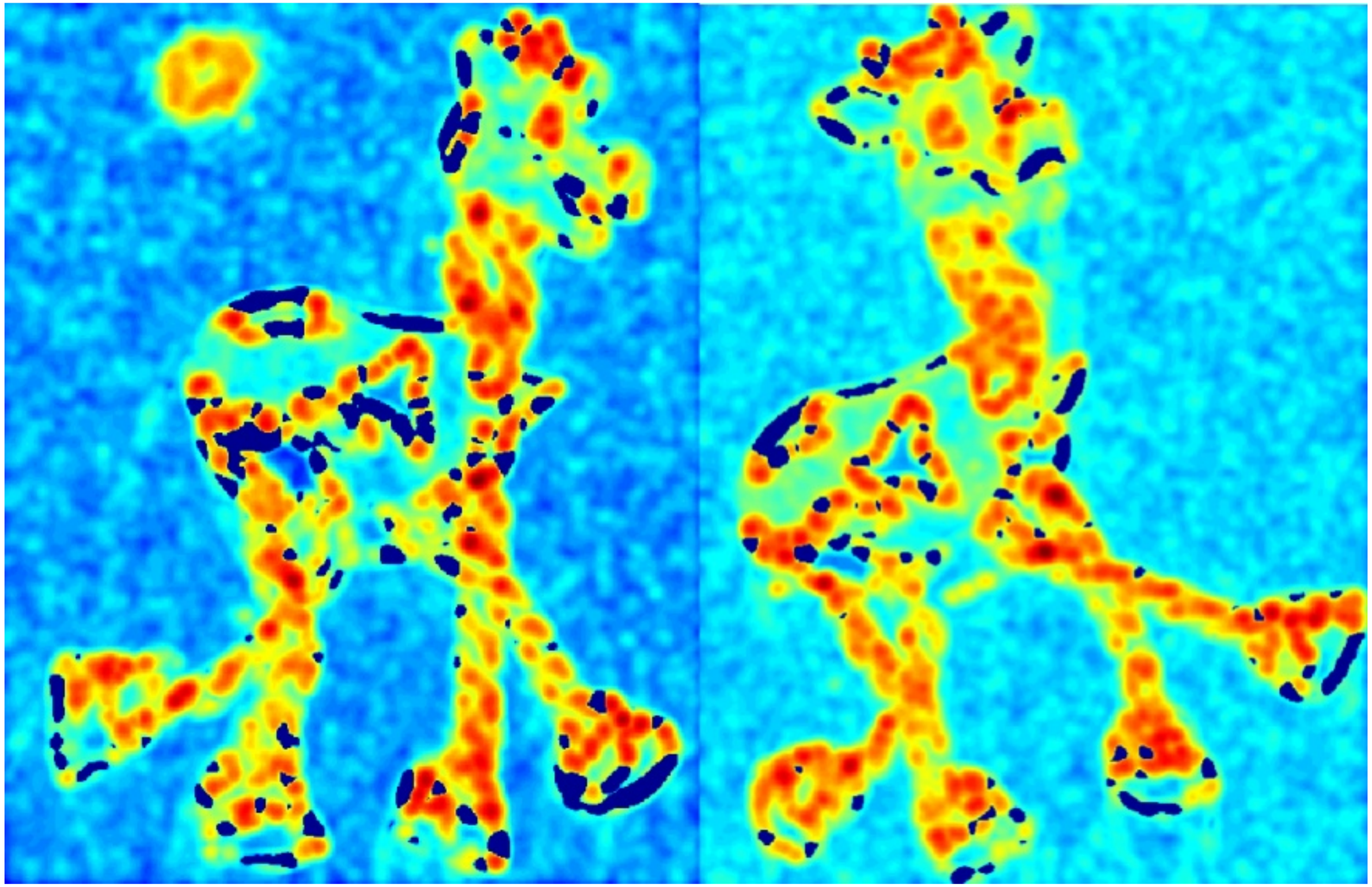
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Harris detector example



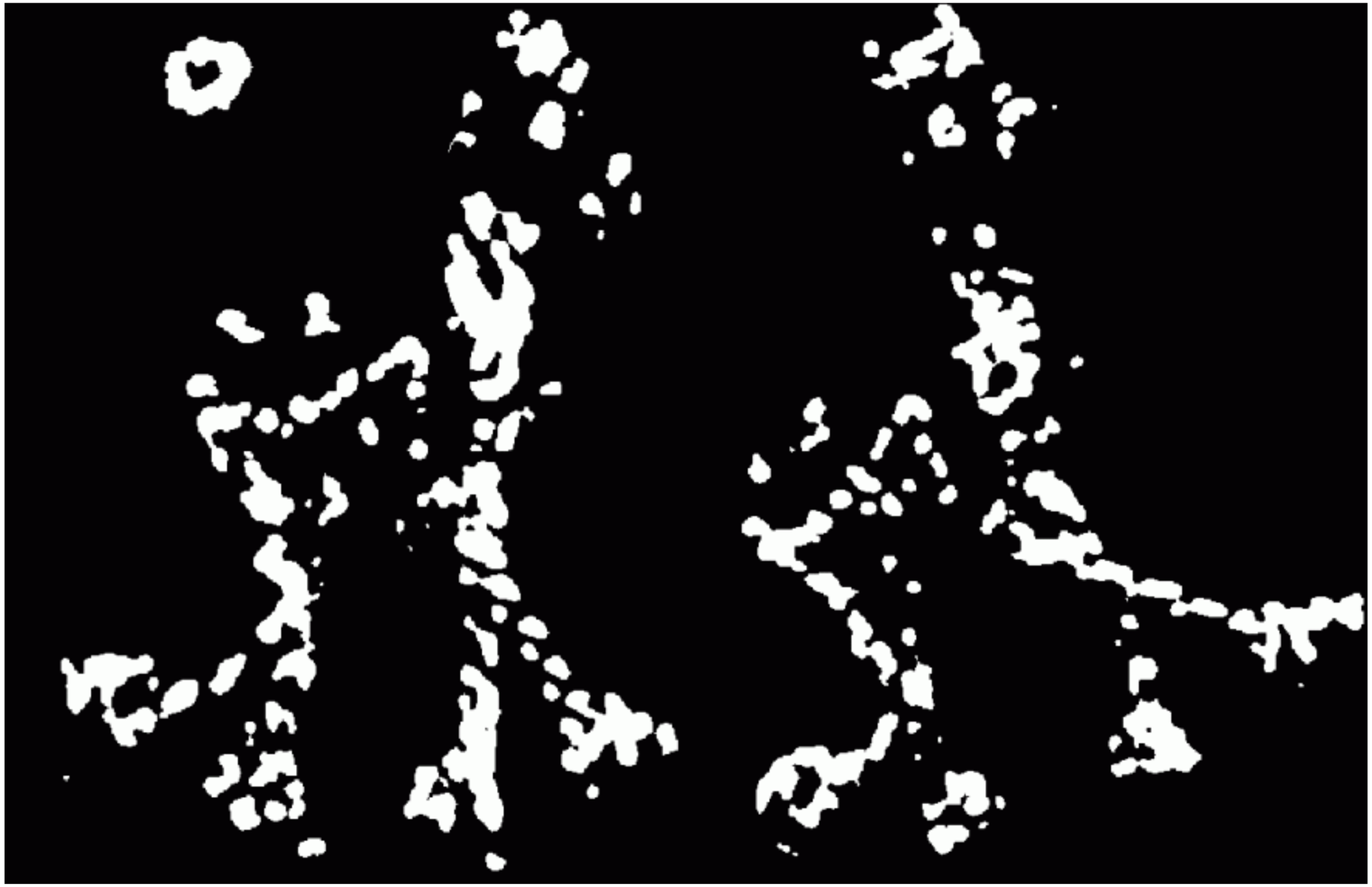
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

f value (red high, blue low)



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Threshold (f > value)



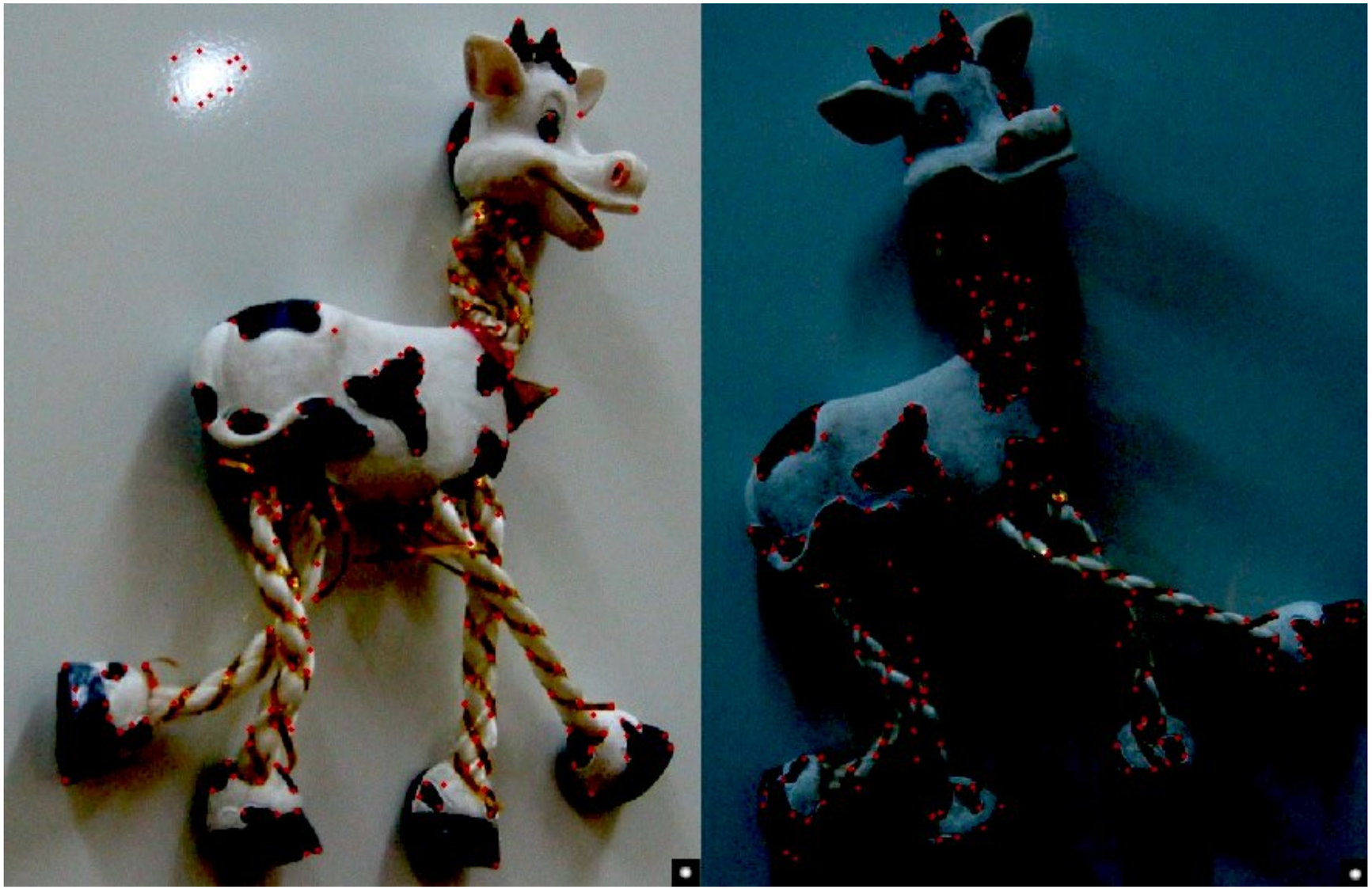
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Find local maxima of f (non-max suppression)



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

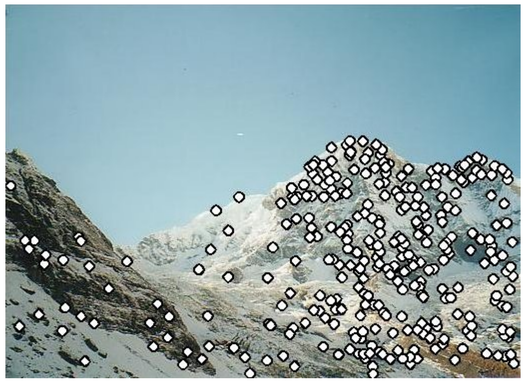
Harris features (in red)



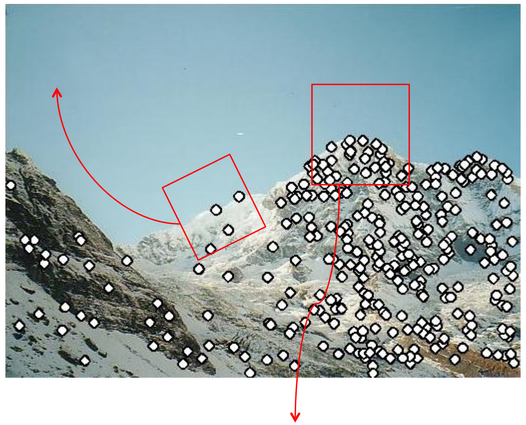
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Local features: main components

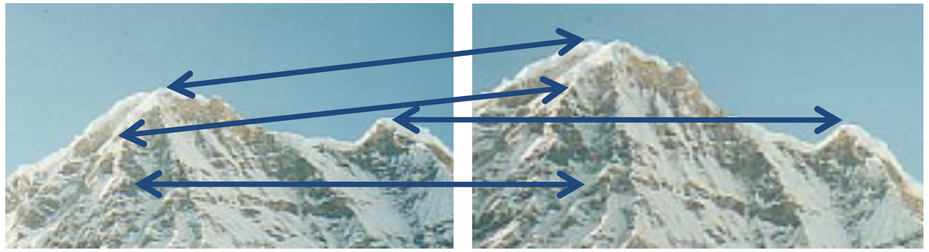
1) Detection: Identify the interest points



2) **Description: Extract vector feature descriptor surrounding each interest point.**



3) Matching: Determine correspondence between descriptors in two views

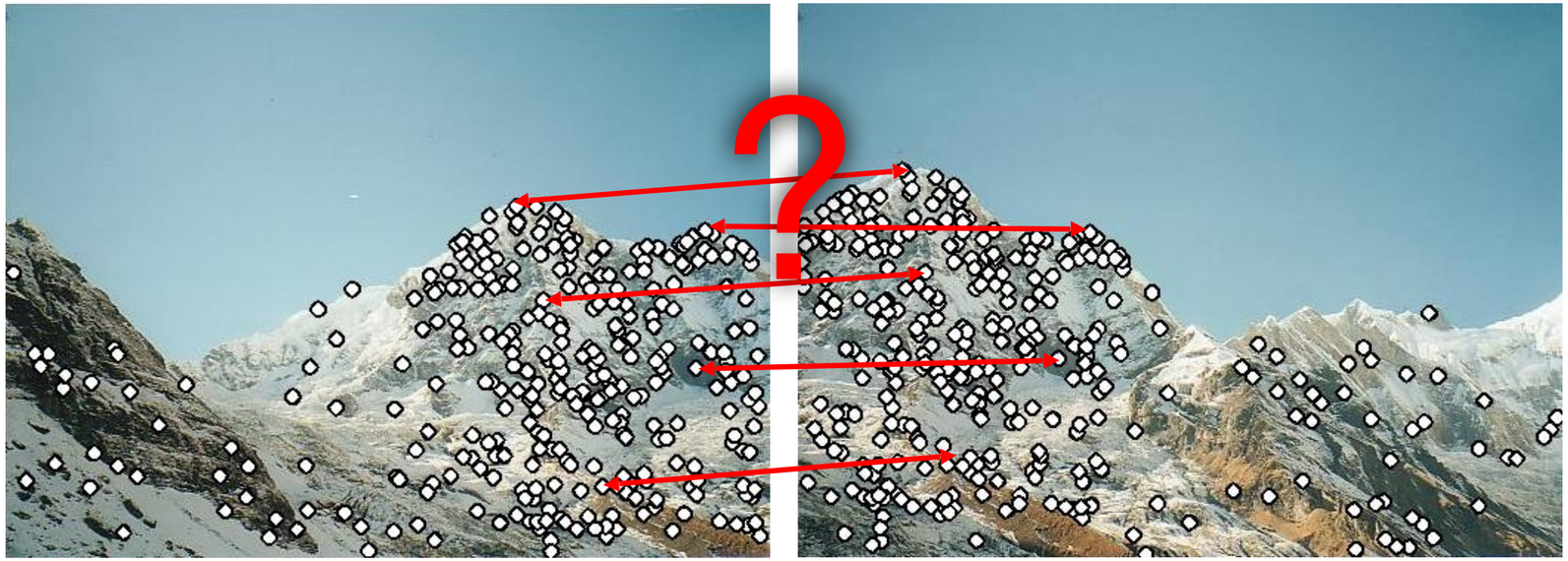


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Kristen Grauman

Feature descriptors

We know how to detect good points
Next question: **How to match them?**

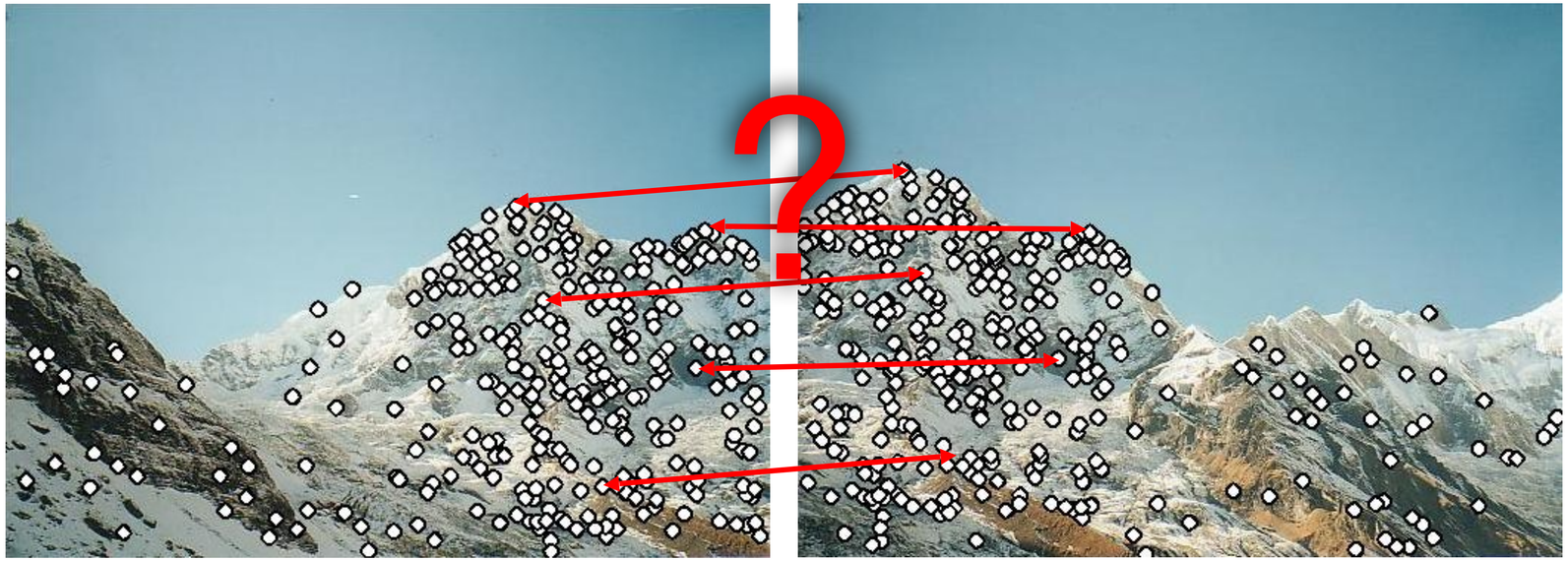


Answer: Come up with a *descriptor* for each point, find similar descriptors between the two images

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Feature descriptors

We know how to detect good points
Next question: **How to match them?**



Lots of possibilities

- Simple option: match square windows around the point
- State of the art approach: SIFT
  - David Lowe, UBC http://www.cs.ubc.ca/~lowe/keypoints/

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Invariance vs. discriminability

- Invariance:
  - Descriptor shouldn’t change even if image is transformed

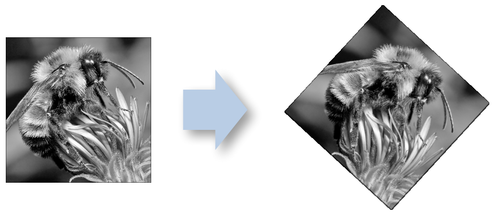
- Discriminability:
  - Descriptor should be highly unique for each point

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

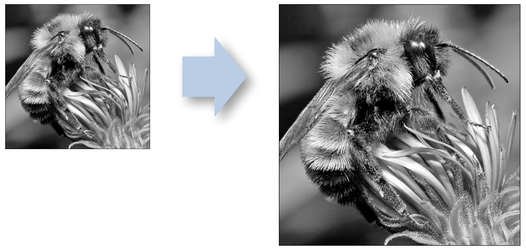
Image transformations revisited

- Geometric

Rotation

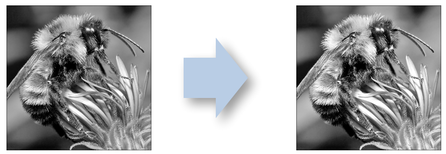


Scale



- Photometric

Intensity change



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Invariant descriptors

- We looked at invariant / equivariant **detectors**
- Most feature descriptors are also designed to be invariant to:
  - Translation, 2D rotation, scale

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Invariant descriptors

- We looked at invariant / equivariant **detectors**
- Most feature descriptors are also designed to be invariant to:
  - Translation, 2D rotation, scale
- They can usually also handle
  - Limited 3D rotations (SIFT works up to about 60 degrees)
  - Limited affine transforms (some are fully affine invariant)
  - Limited illumination/contrast changes

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
2. Design an invariant feature descriptor

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
2. Design an invariant feature descriptor
   - Simplest descriptor: a single 0
     - What's this invariant to?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
2. Design an invariant feature descriptor
   - Simplest descriptor: a single 0
     - What's this invariant to?
   - Next simplest descriptor: a square, axis-aligned 5x5 window of pixels
     - What's this invariant to?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
2. Design an invariant feature descriptor
   - Simplest descriptor: a single 0
     - What's this invariant to?
   - Next simplest descriptor: a square, axis-aligned 5x5 window of pixels
     - What's this invariant to?
   - Let's look at some better approaches…

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Rotation invariance for feature descriptors

- Find dominant orientation of the image patch
  - E.g., given by $\mathbf{x}_{\mathbf{max}}$, the eigenvector of **H** corresponding to λ_{max} (the *larger* eigenvalue)
  - Or (better) simply the orientation of the (smoothed) gradient
  - Rotate the patch according to this angle



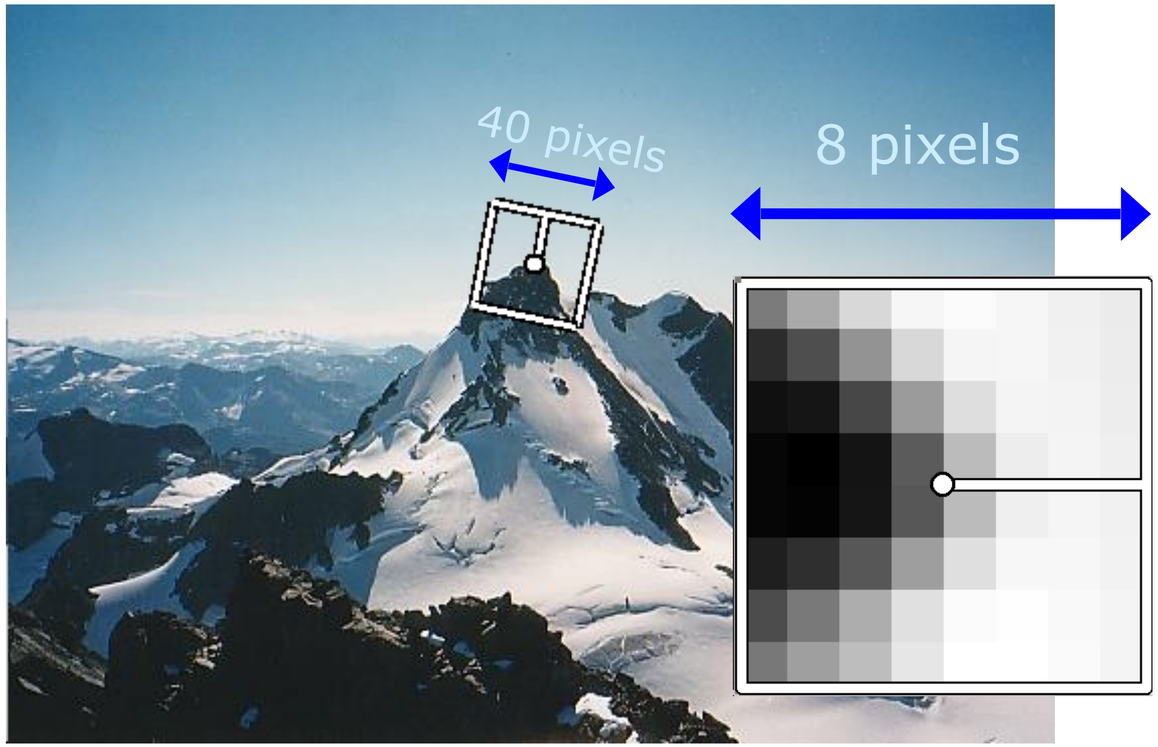
Figure by Matthew Brown

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window (why?)



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Adapted from slide by Matthew Brown

Detections at multiple scales

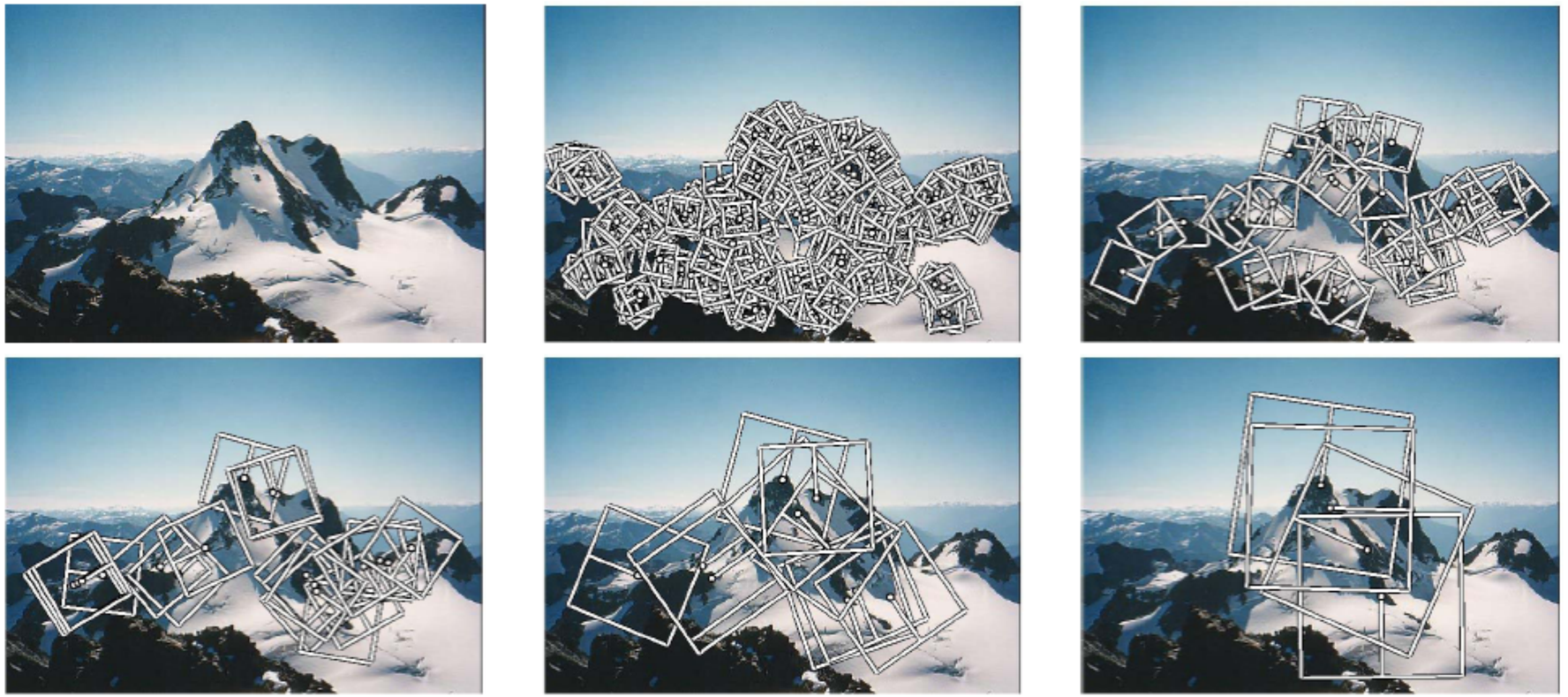


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Scale Invariant Feature Transform

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations
- Shift the bins so that the biggest one is first

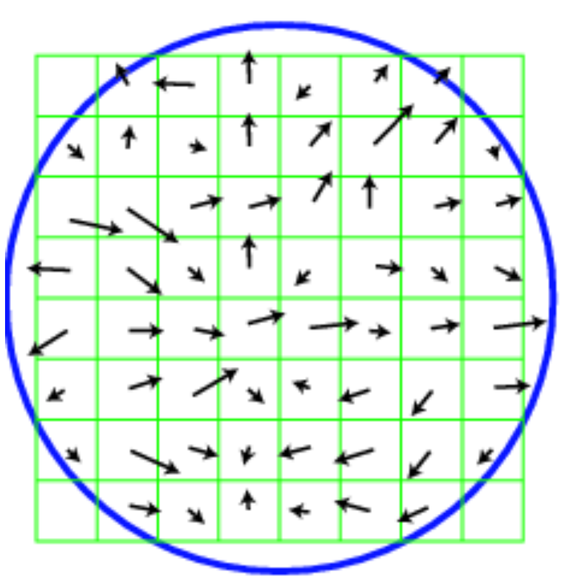
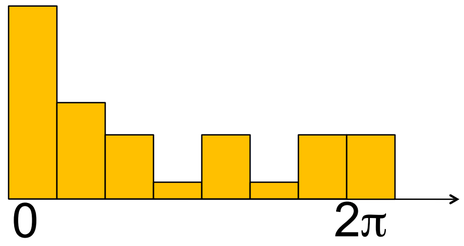
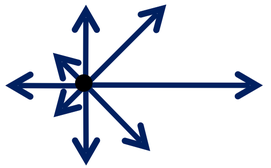


Image gradients



angle histogram



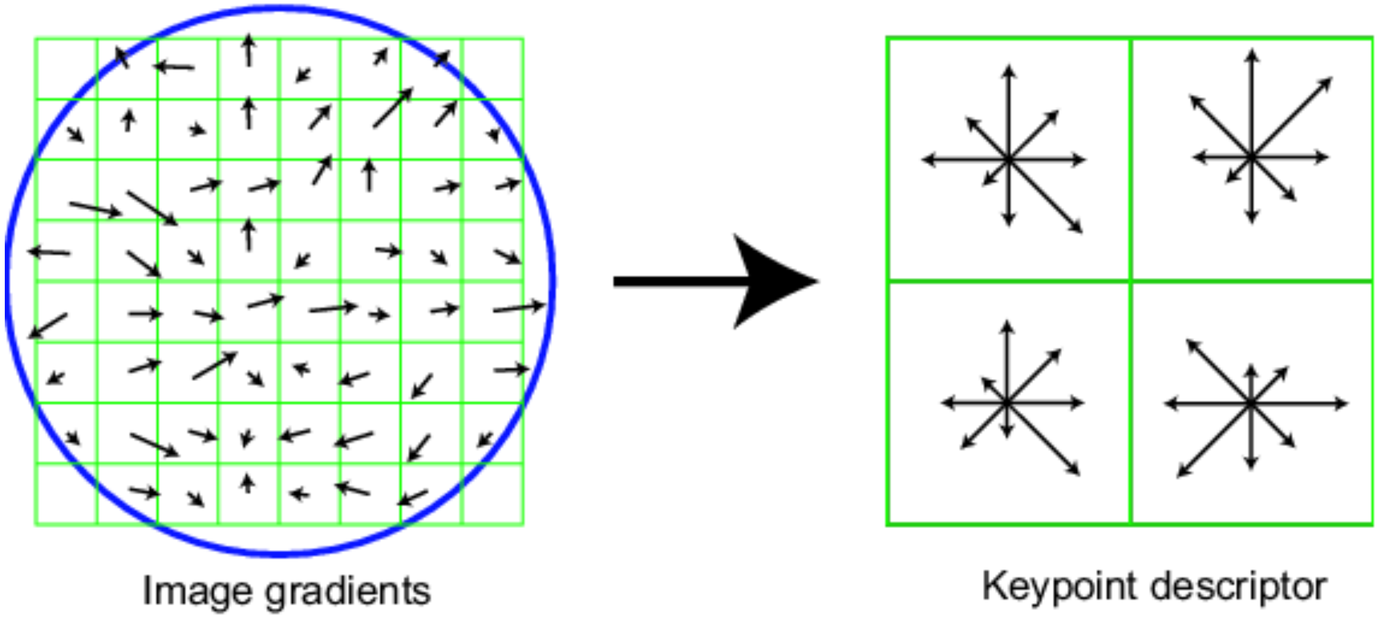
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Adapted from slide by David Lowe

SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Adapted from slide by David Lowe

Properties of SIFT

Extraordinarily robust matching technique

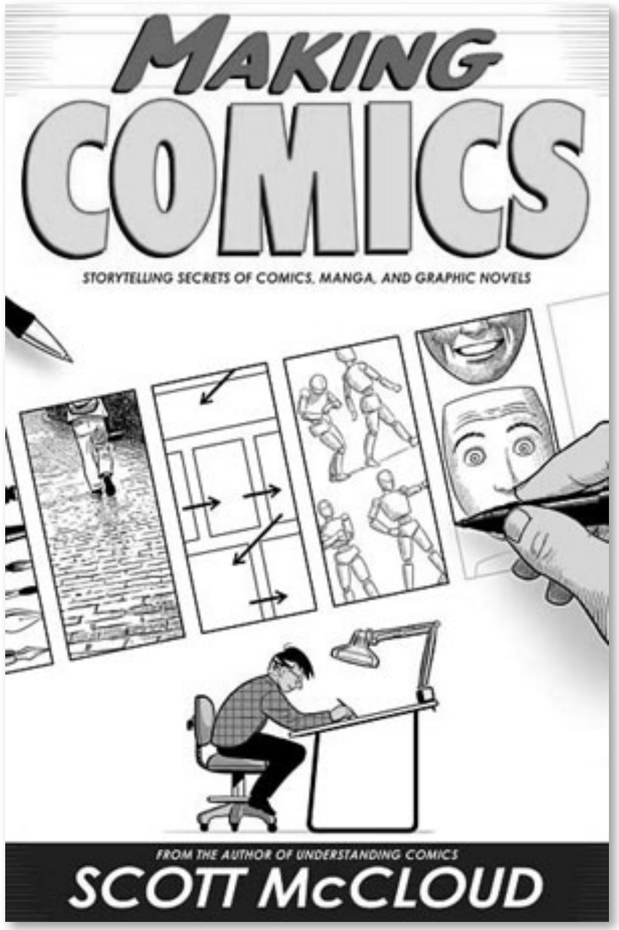
- Can handle changes in viewpoint (up to about 60 degree out of plane rotation)
- Can handle significant changes in illumination (sometimes even day vs. night (below))
- Pretty fast—hard to make real-time, but can run in <1s for moderate image sizes
- Lots of code available





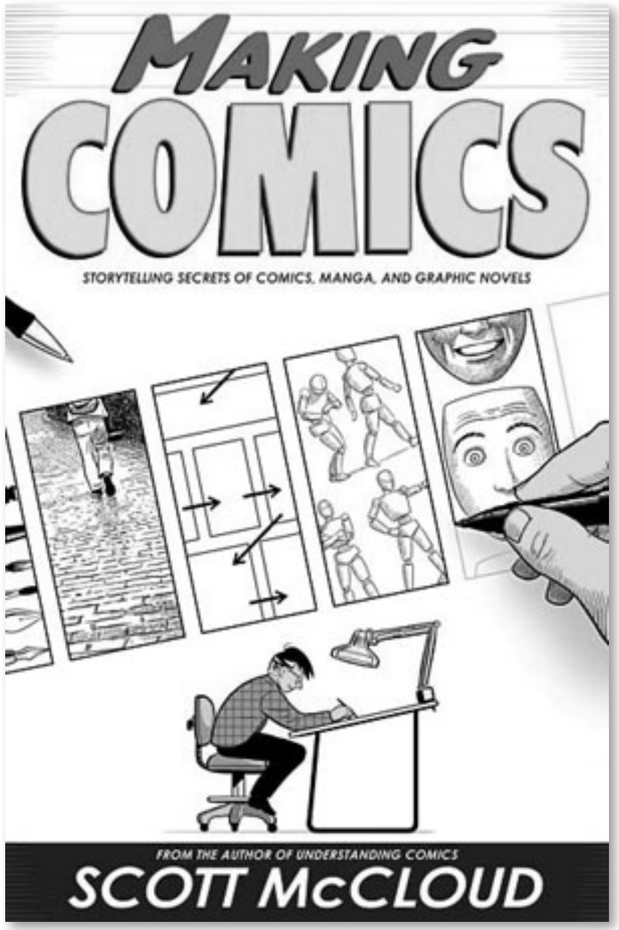
Slide credit: Noah Snavely, CS 670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

SIFT Example

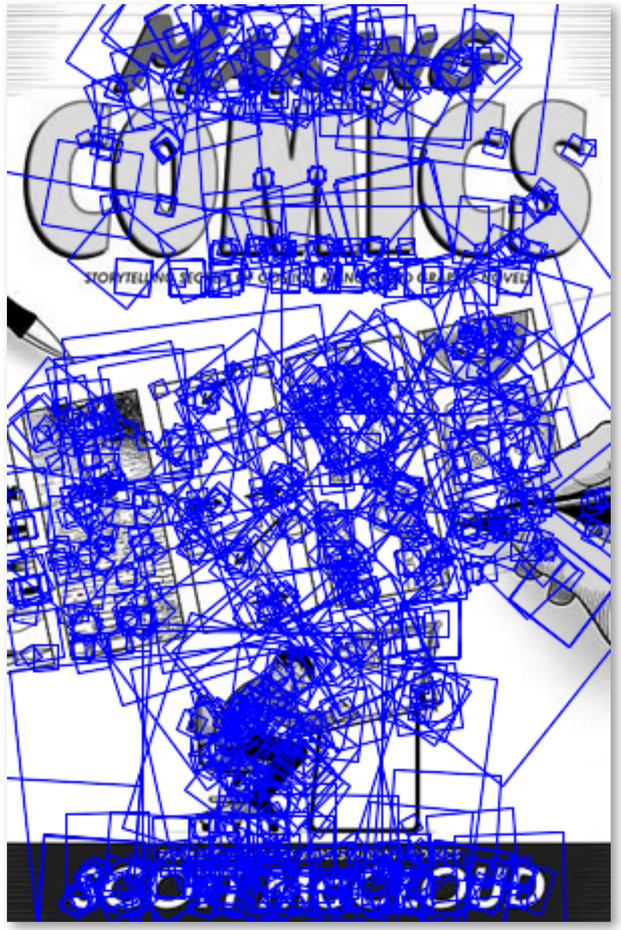


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

SIFT Example



sift

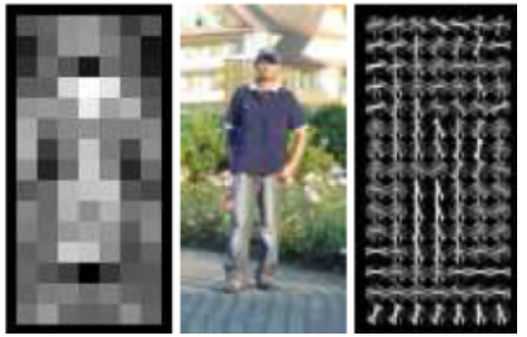


868 SIFT features

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Other descriptors

- HOG: Histogram of Gradients (HOG)
  - Dalal/Triggs
  - Sliding window, pedestrian detection



- FREAK: Fast Retina Keypoint
  - Perceptually motivated
  - Can run in real-time; used in Visual SLAM on-device

- LIFT: Learned Invariant Feature Transform
  - Learned via deep learning – along with many other recent features

https://arxiv.org/abs/1603.09114

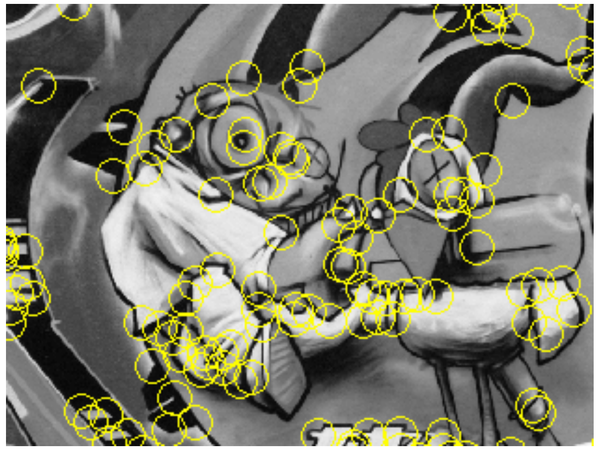
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Questions?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Summary

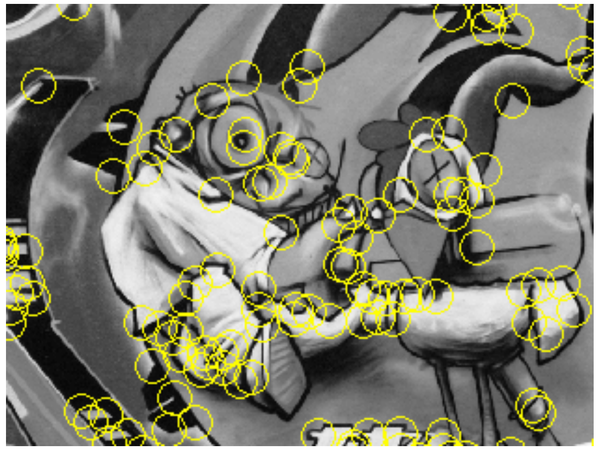
- Keypoint detection: repeatable and distinctive
  - Corners, blobs
  - Harris, DoG



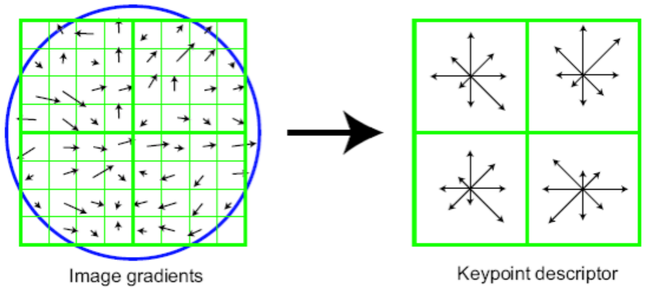
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Summary

- Keypoint detection: repeatable and distinctive
  - Corners, blobs
  - Harris, DoG

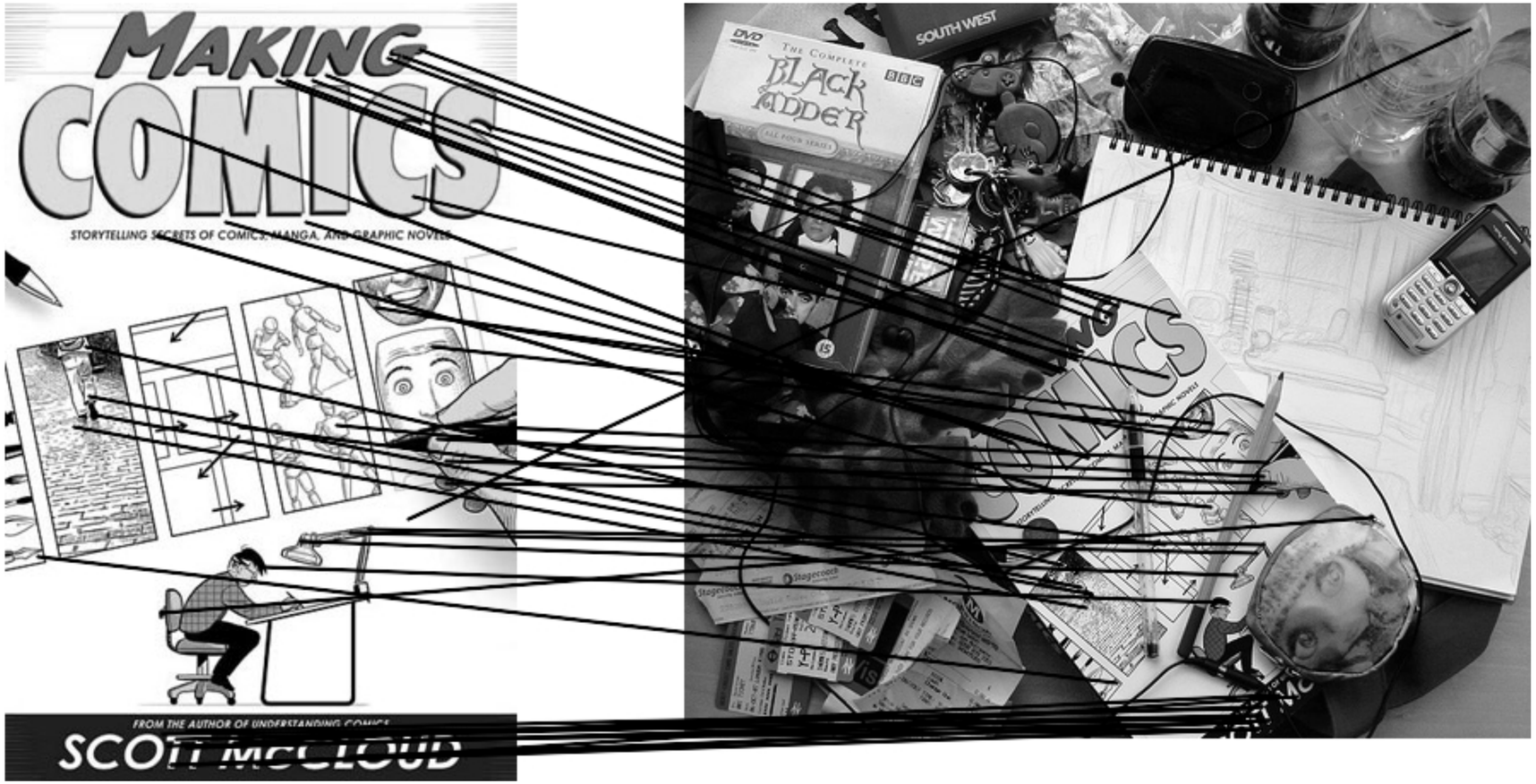


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one



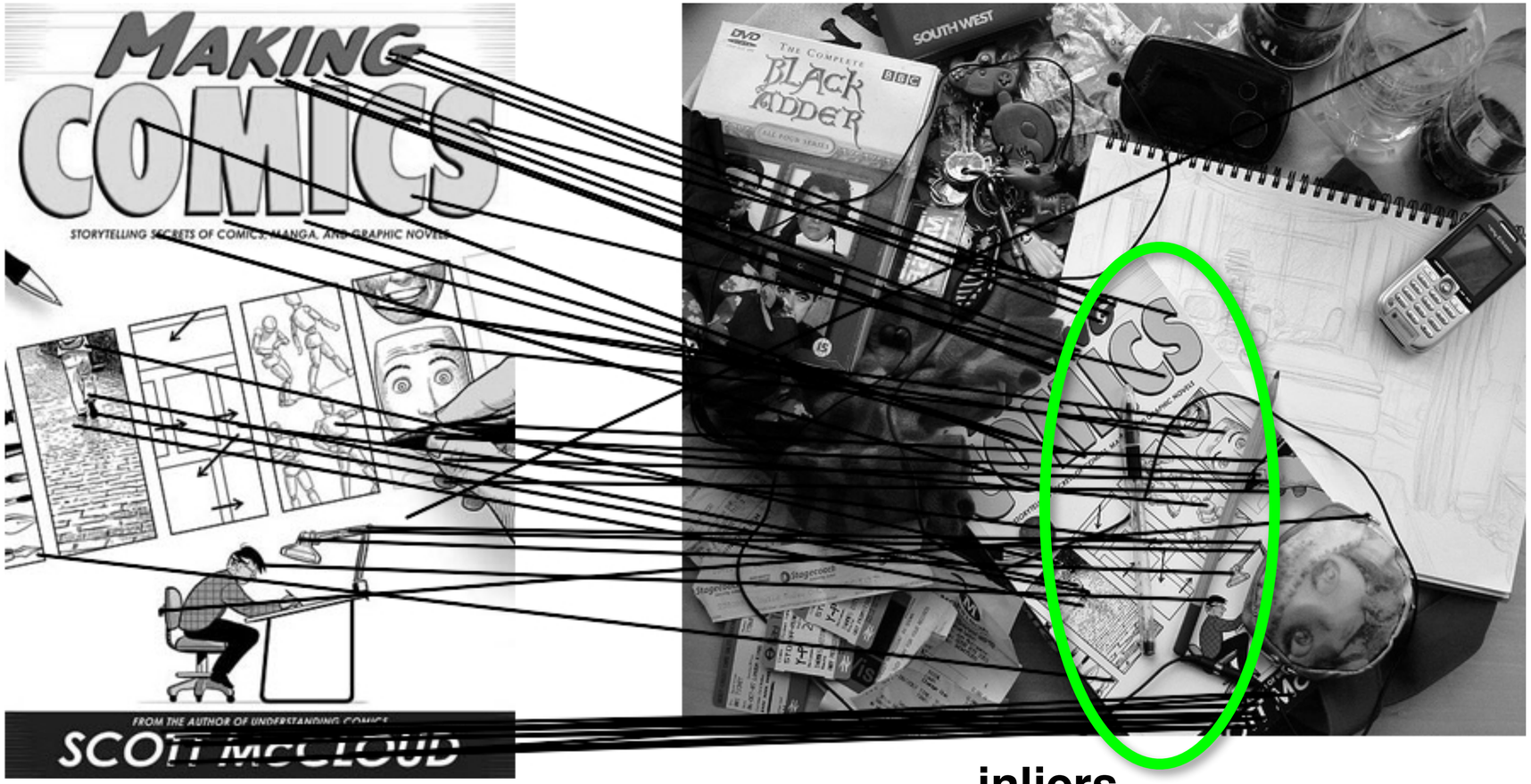
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Outliers



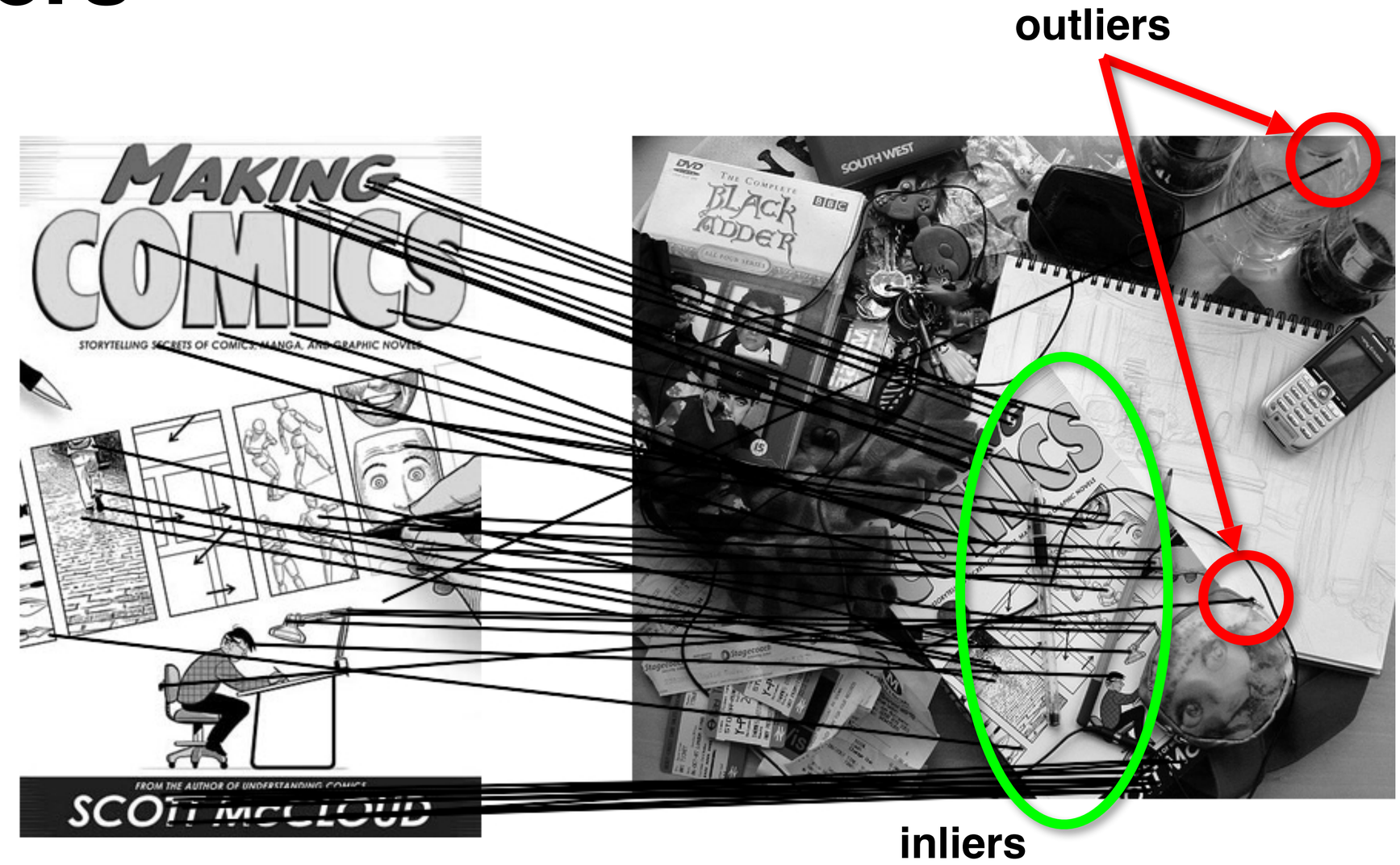
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Outliers



inliers

Outliers



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Robustness

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

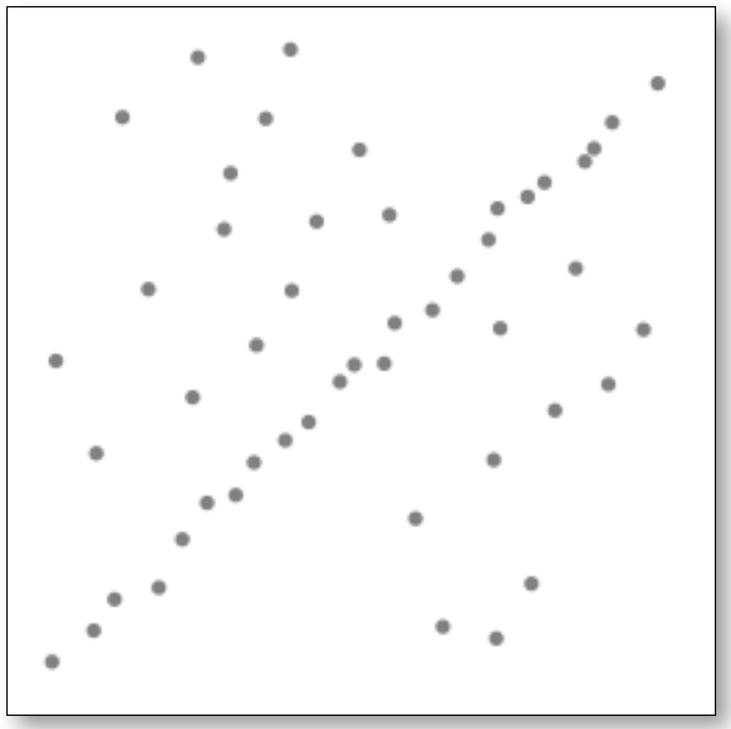
Robustness



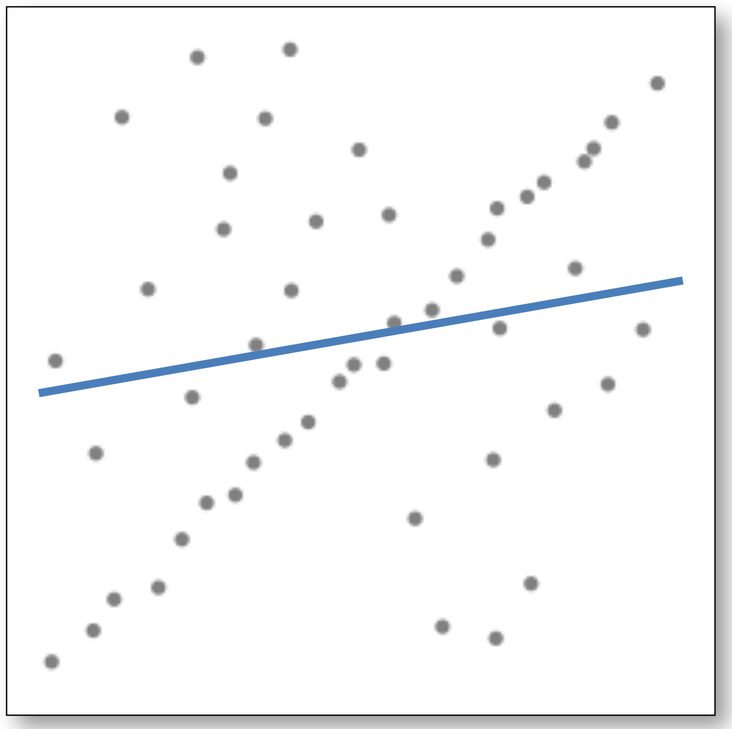
Problem: Fit a line to these datapoints

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Robustness



Problem: Fit a line to these datapoints

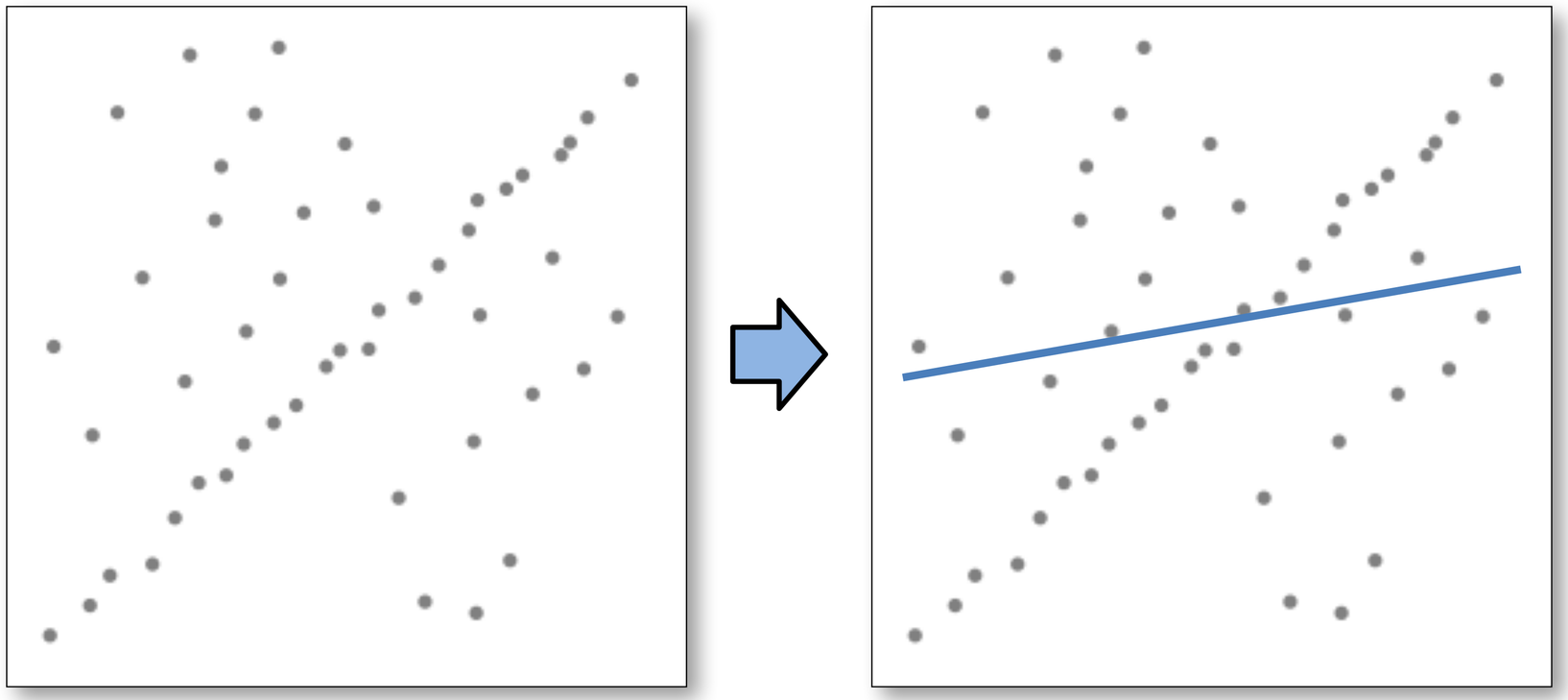


Least squares fit

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Robustness

- Let’s consider the problem of linear regression



Problem: Fit a line to these datapoints

Least squares fit

- How can we fix this?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

We need a better cost function…

- Suggestions?

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Idea

- Given a hypothesized line

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Idea

- Given a hypothesized line
- Count the number of points that “agree” with the line

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Idea

- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

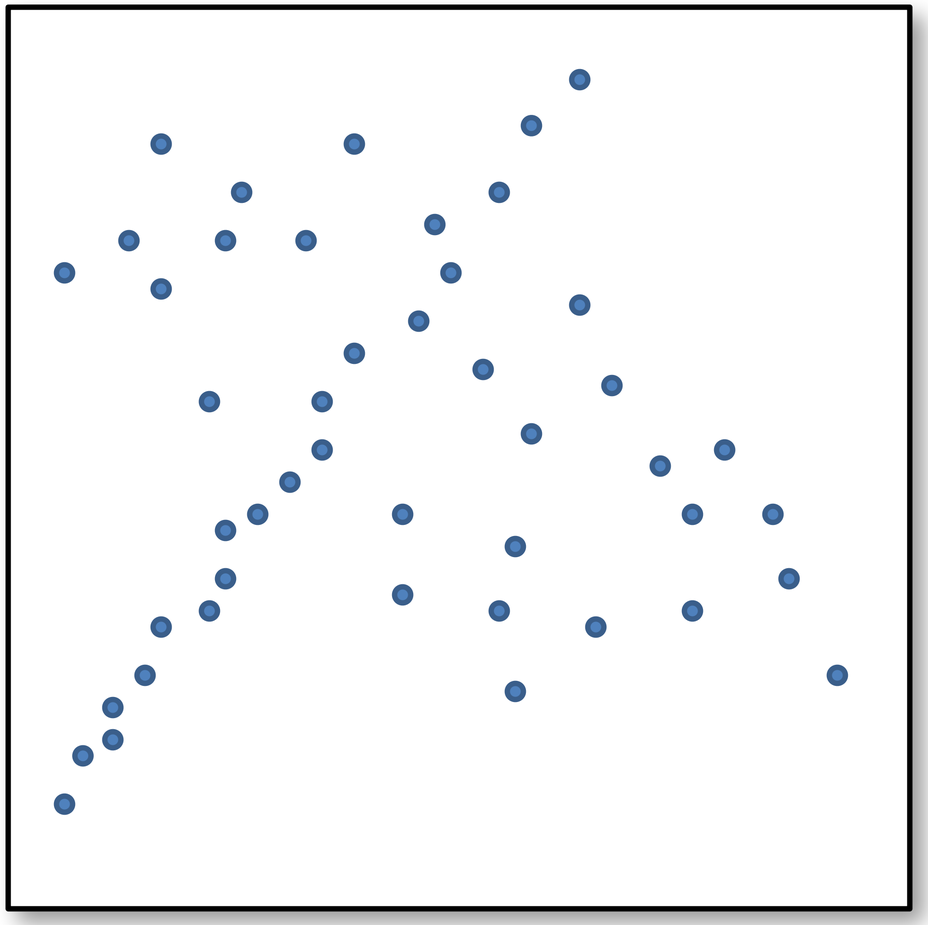
Idea

- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

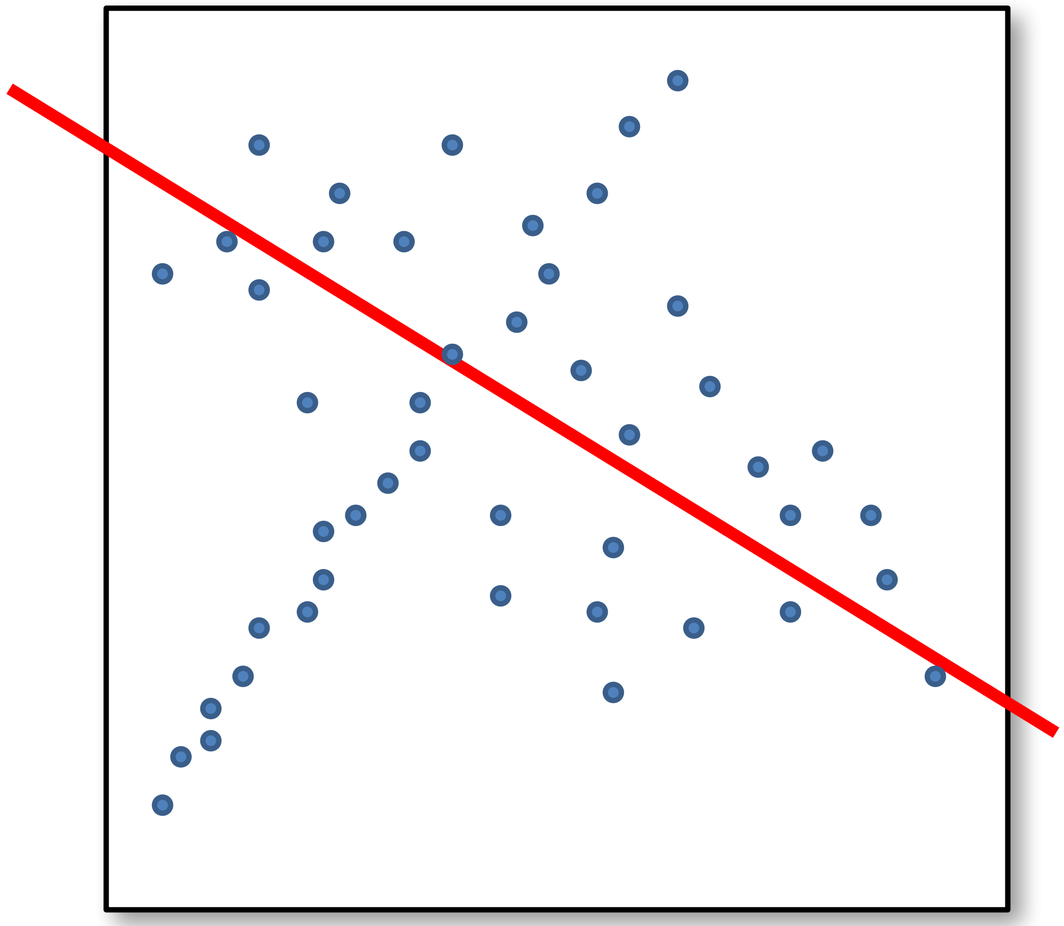
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



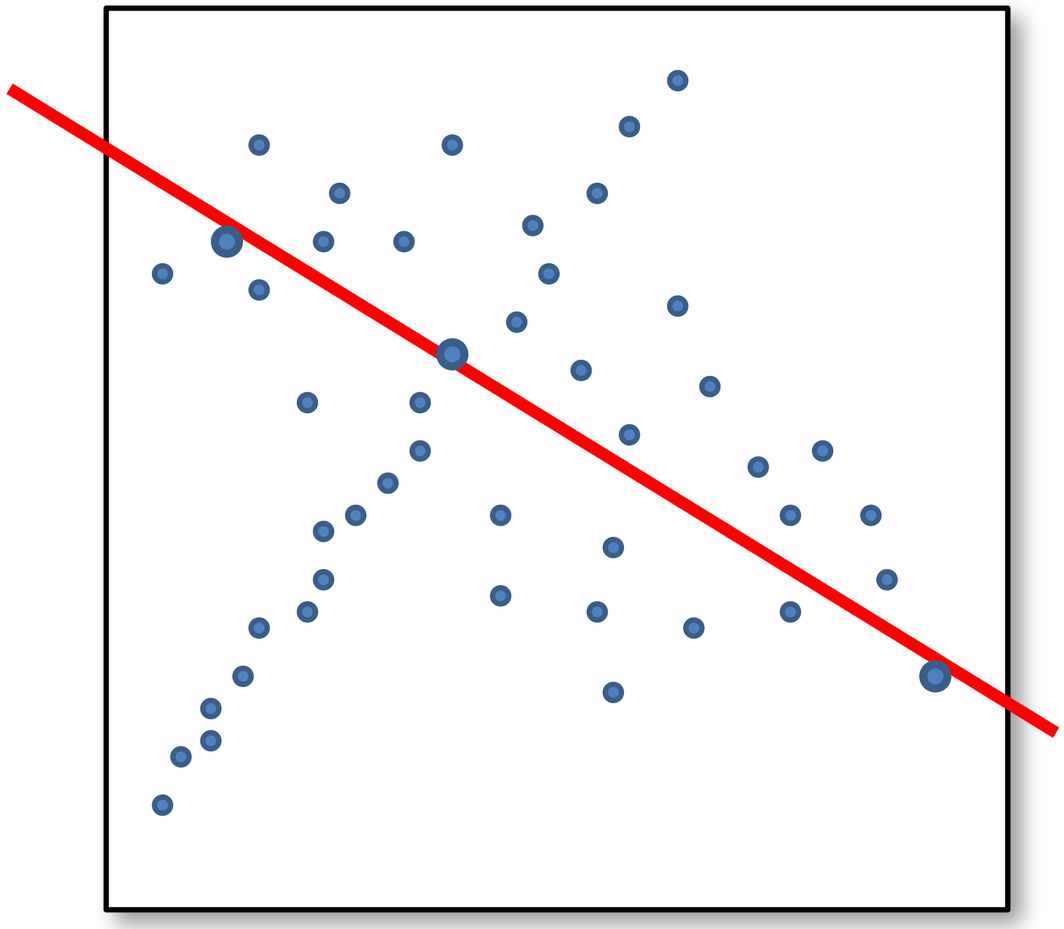
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



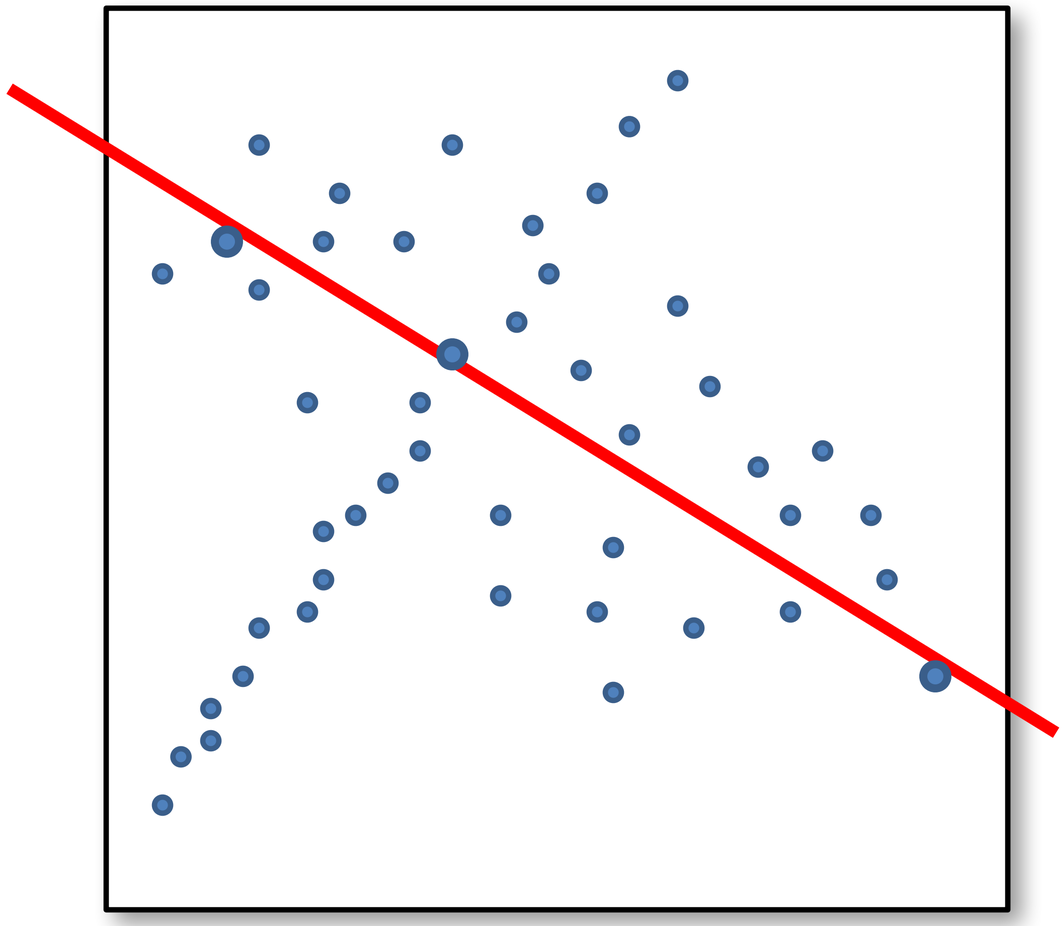
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

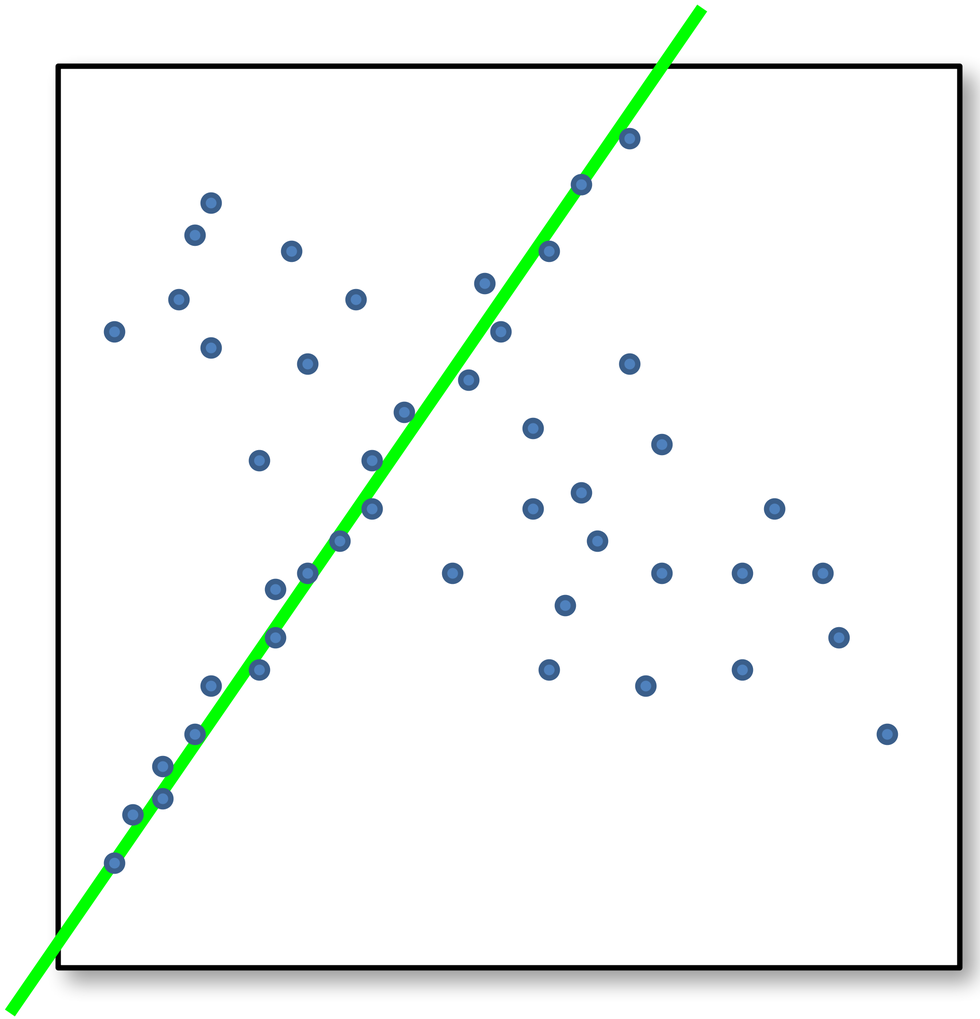
Counting inliers



Inliers: 3

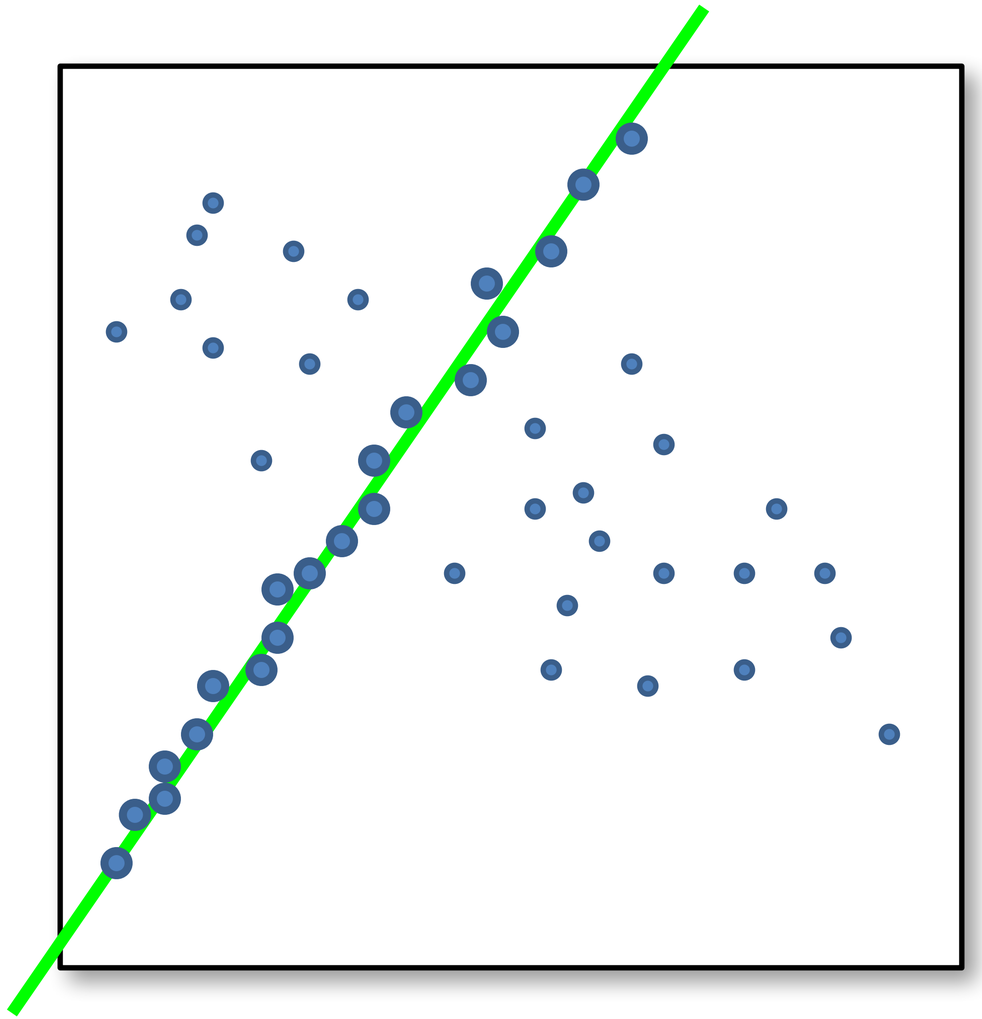
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



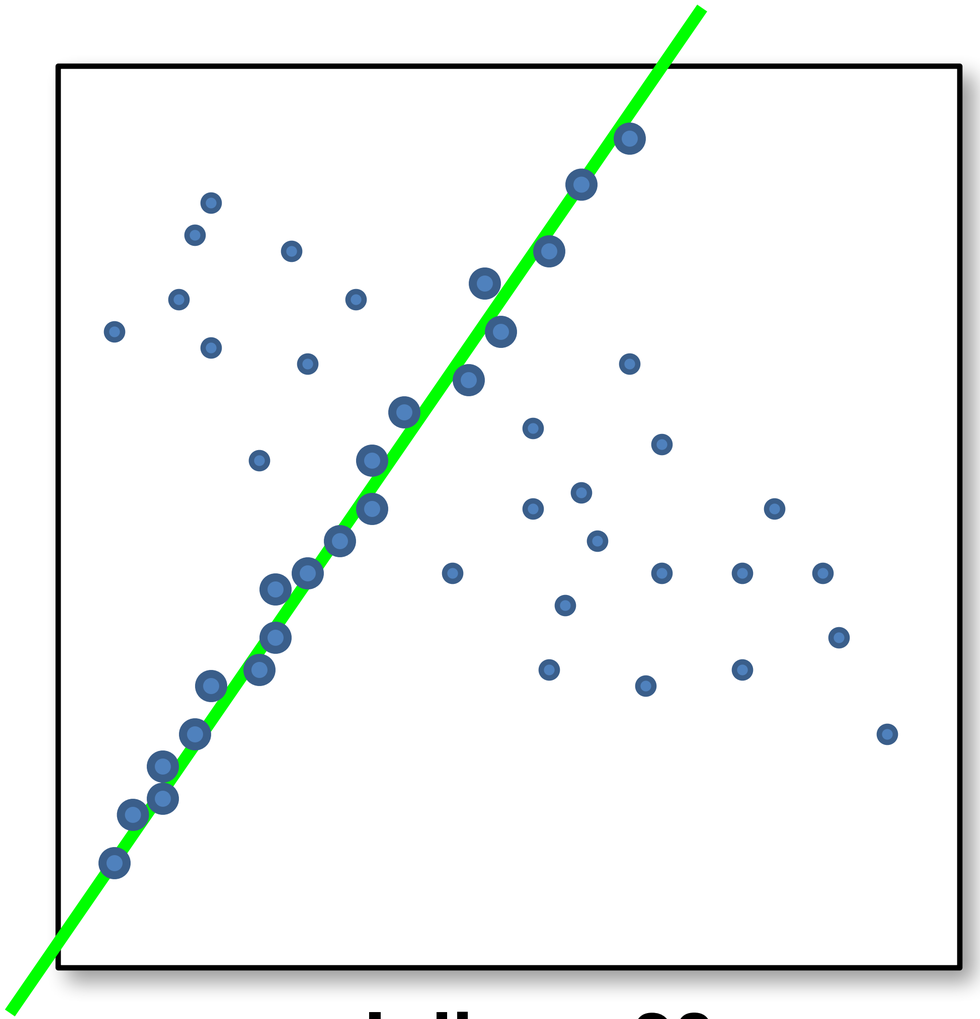
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



Inliers: 20

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How do we find the best line?

- Unlike least-squares, no simple closed-form solution

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How do we find the best line?

- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

How do we find the best line?

- Unlike least-squares, no simple closed-form solution

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

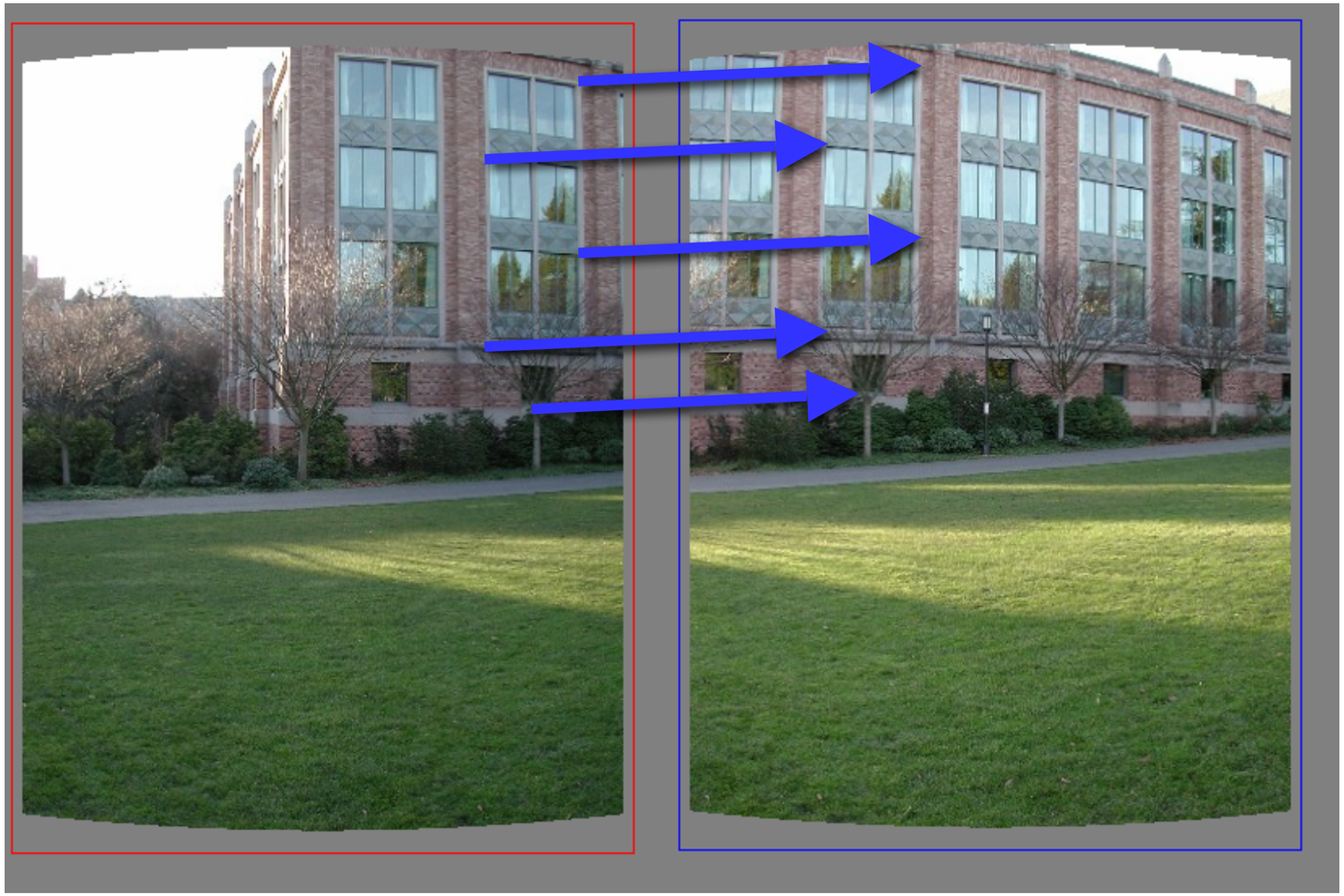
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



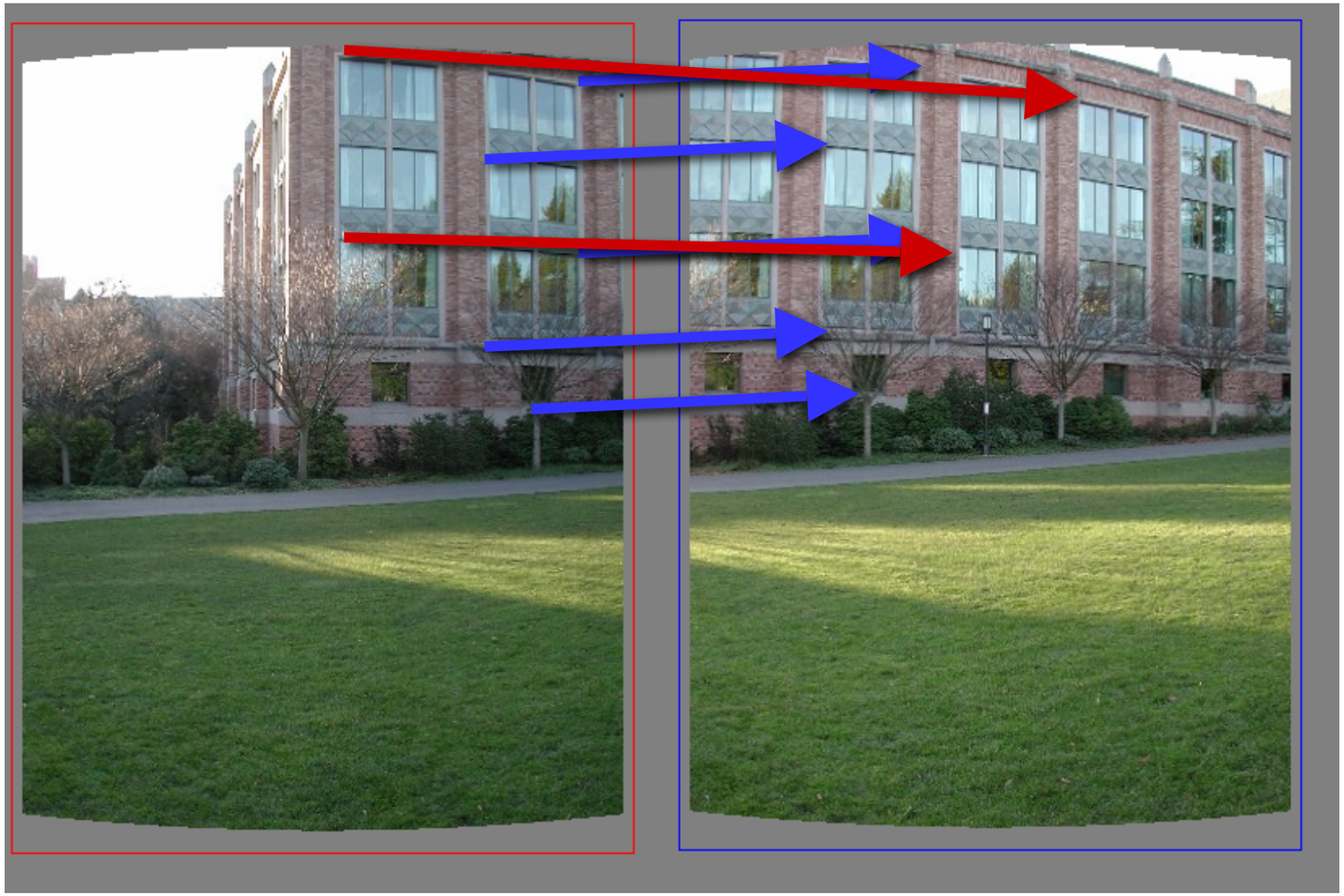
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



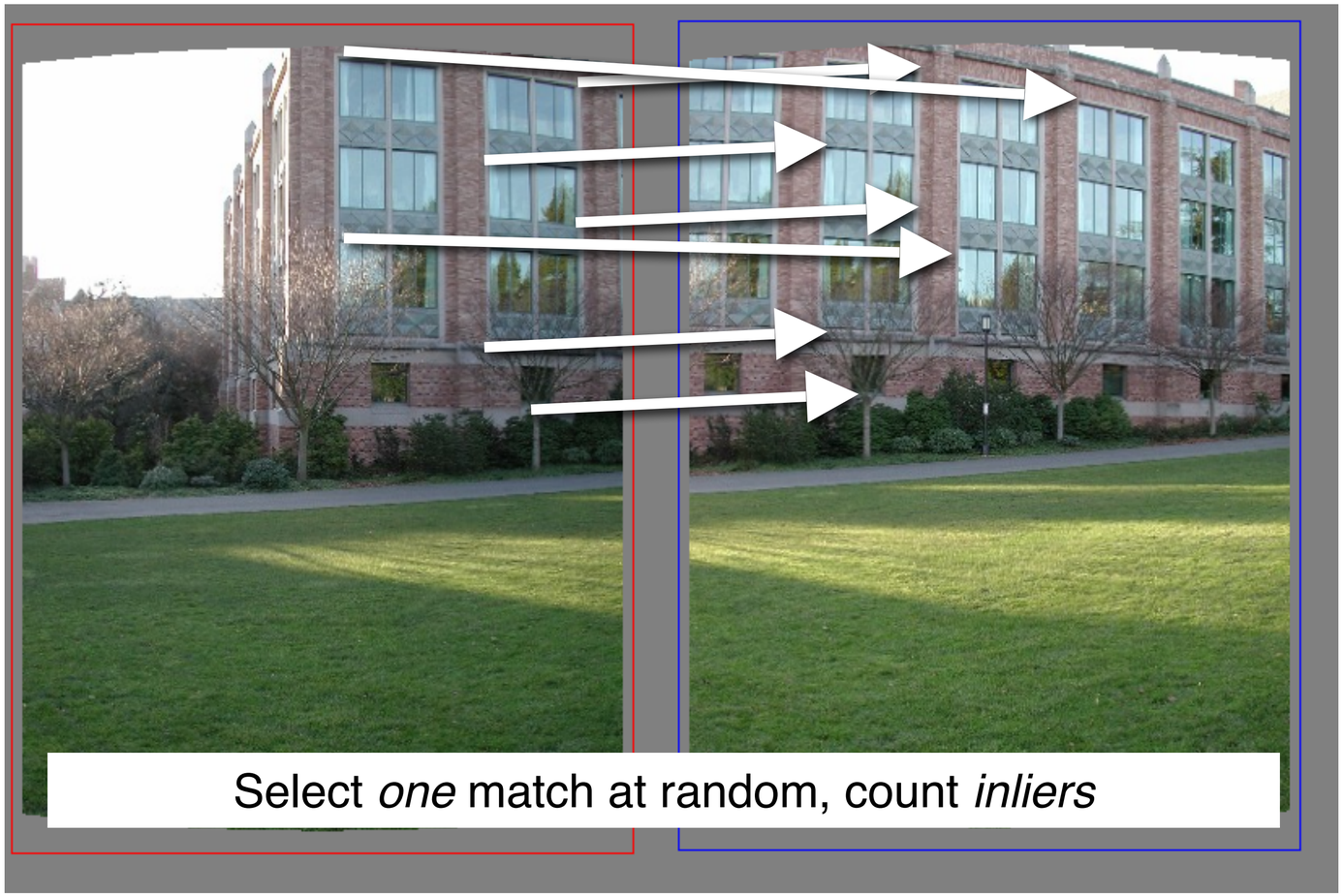
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



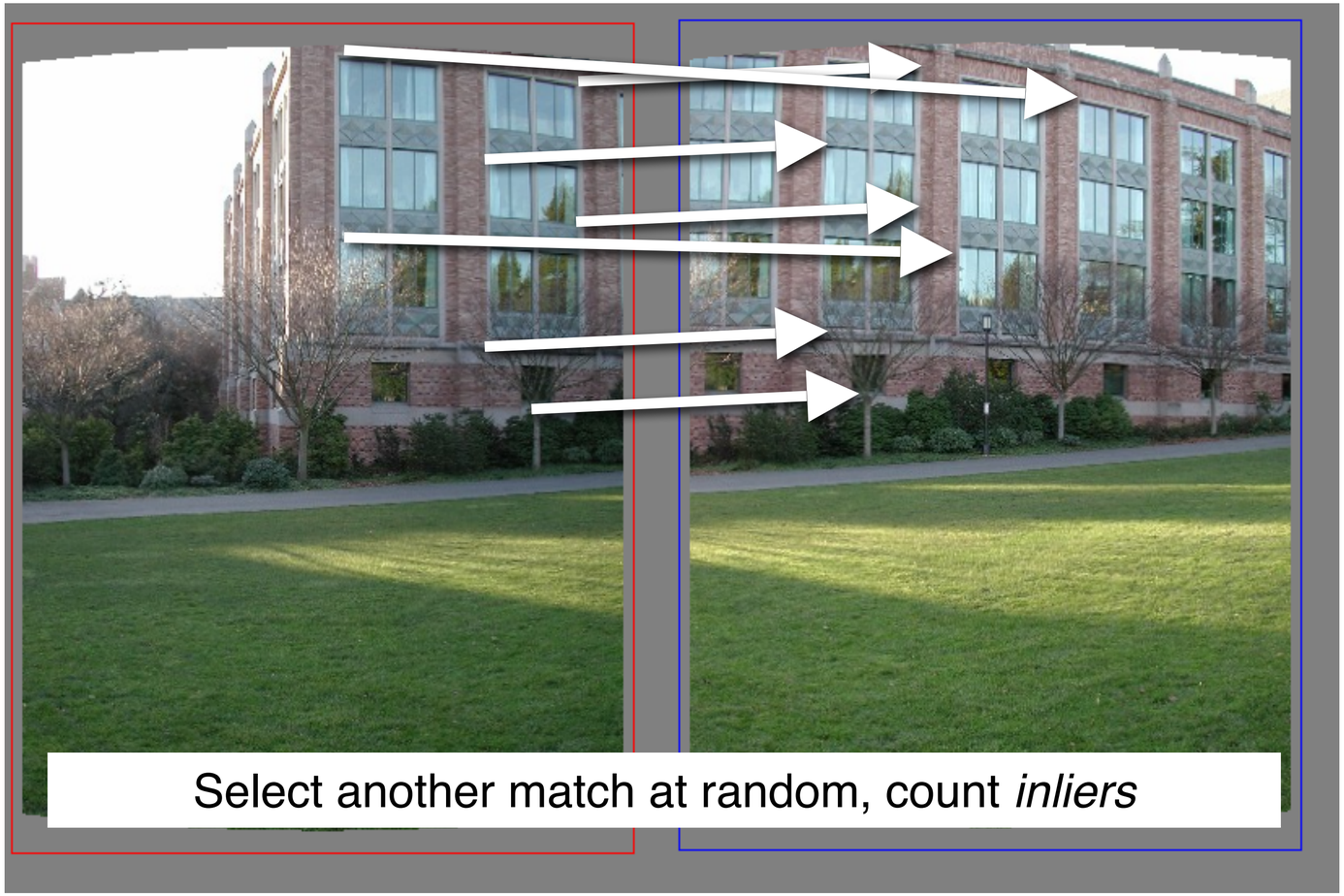
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



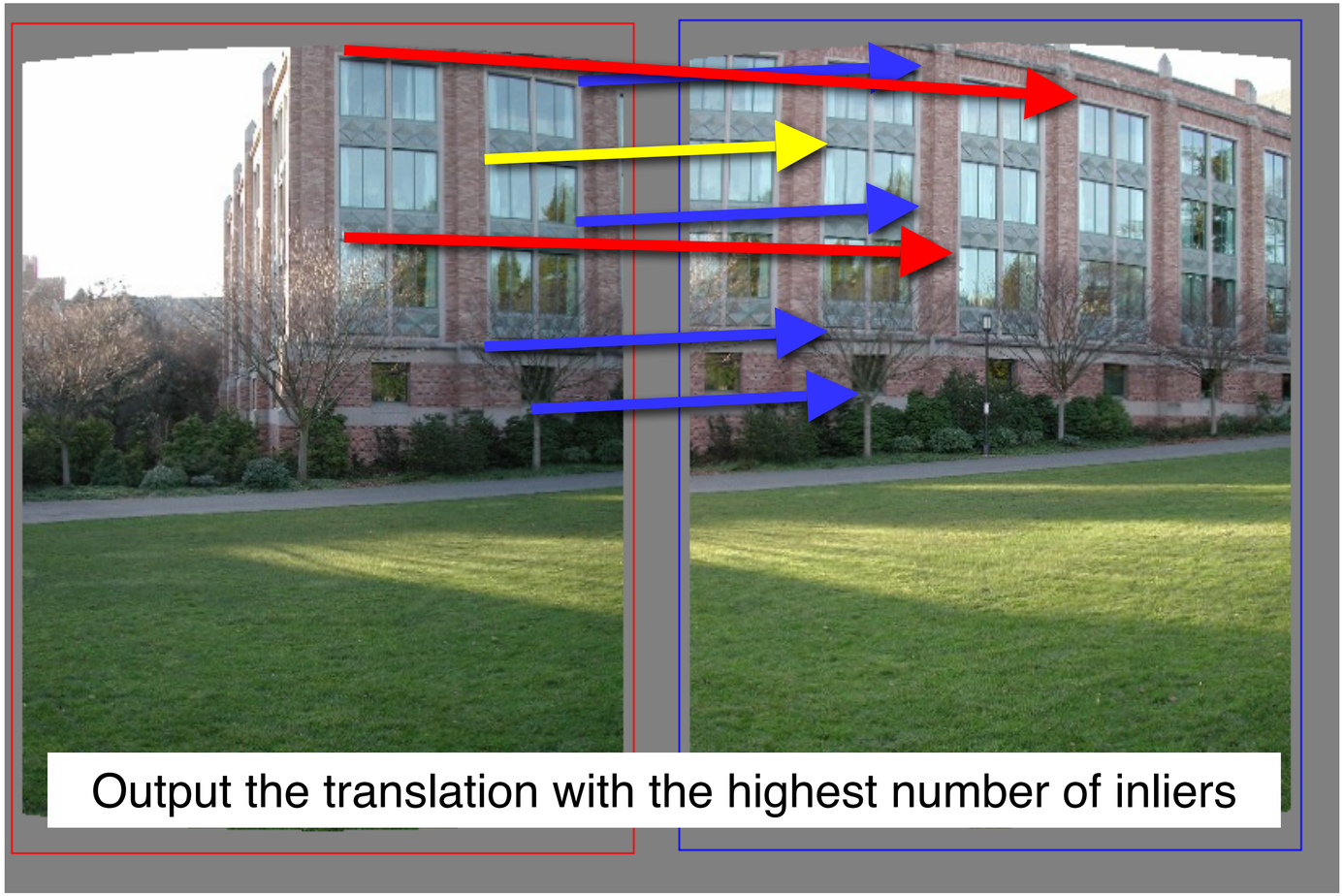
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- Idea:
  - All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other
    - RANSAC only has guarantees if there are < 50% outliers
  - “All good matches are alike; every bad match is bad in its own way.”

    – Tolstoy via Alyosha Efros

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian w/ some standard deviation (e.g. 3 pixels)

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian w/ some standard deviation (e.g. 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee

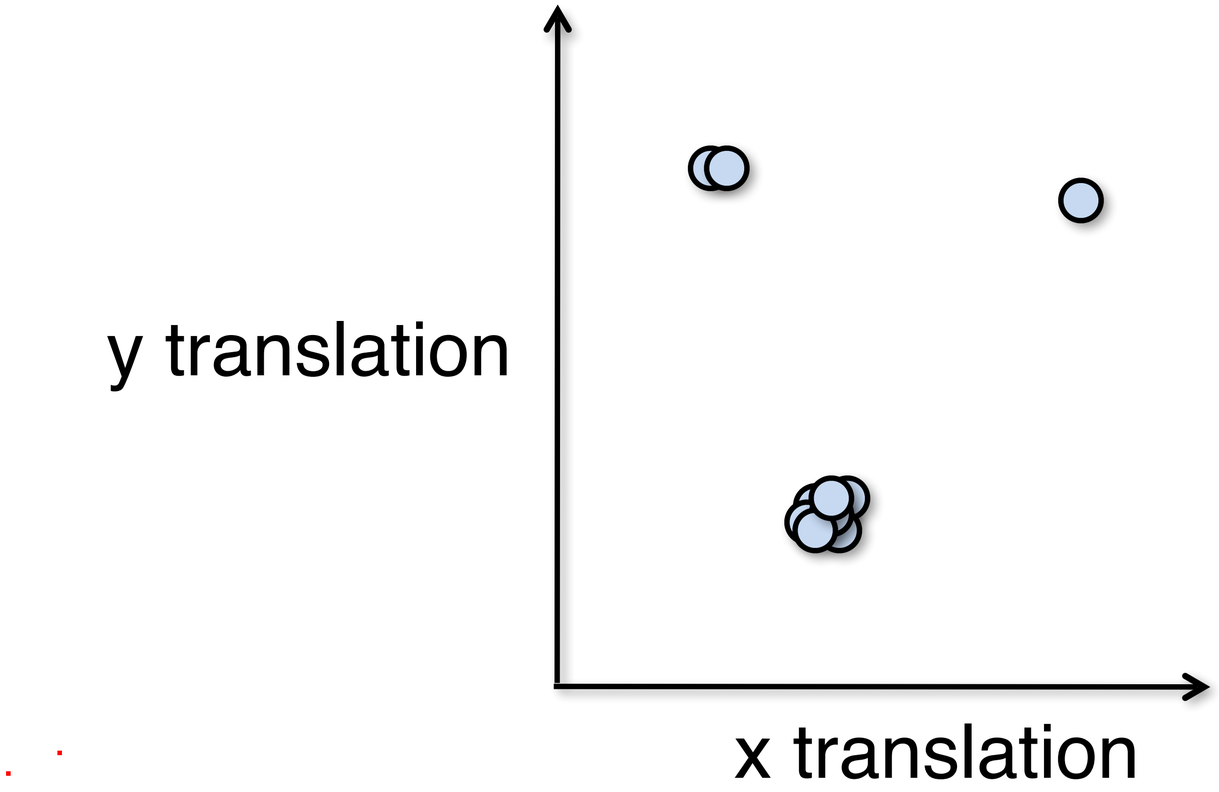
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- **Inlier threshold** related to the amount of noise we expect in inliers
  - Often model noise as Gaussian w/ some standard deviation (e.g. 3 pixels)
- **Number of rounds** related to the percentage of outliers we expect, and the probability of success we'd like to guarantee
  - Suppose there are 20% outliers, and we want to find the correct answer with at least 99% probability
  - How many rounds do we need?

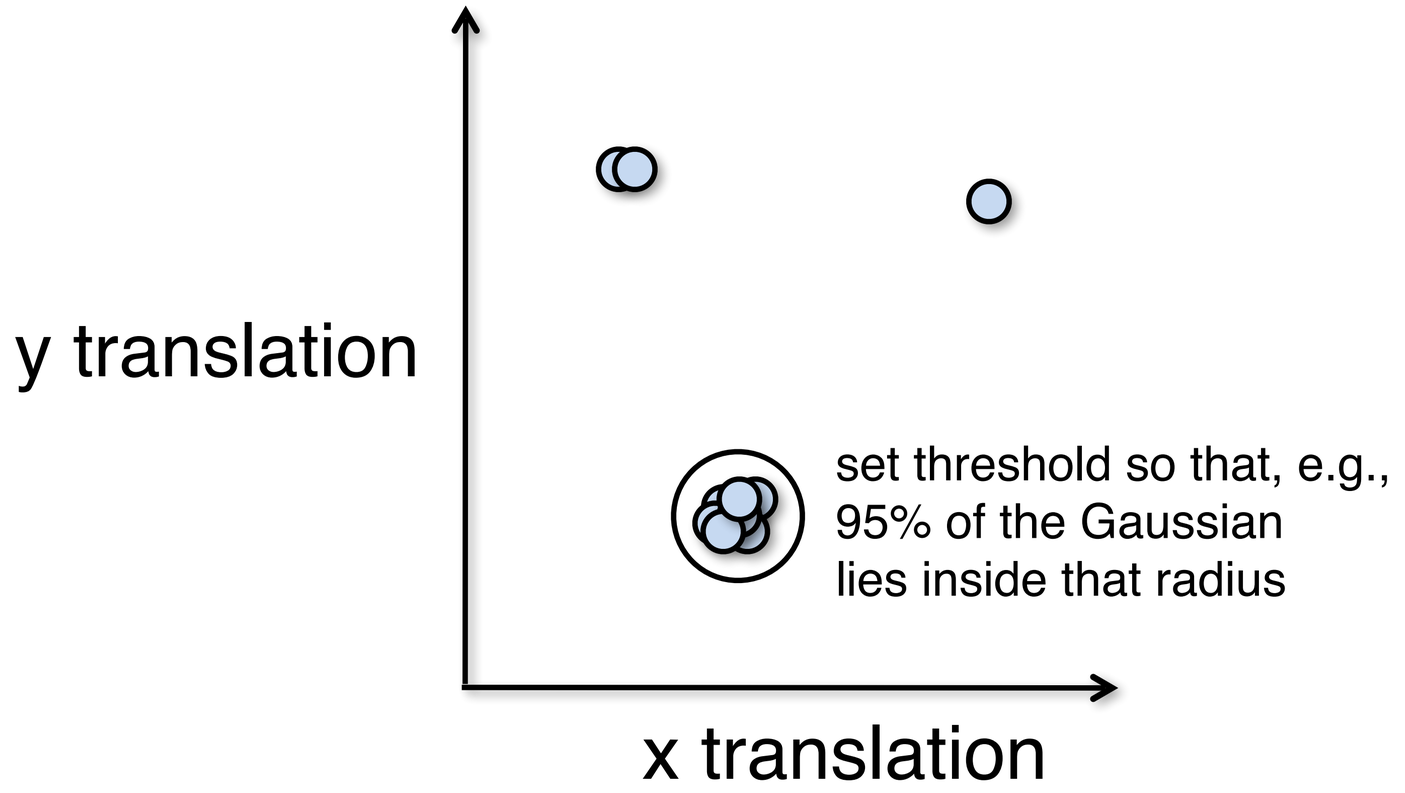
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC: Another view



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC: Another view

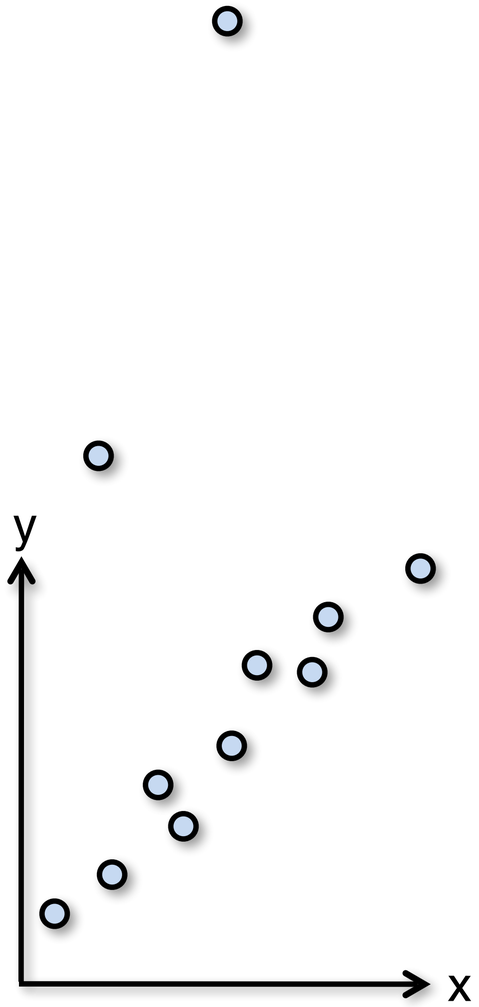


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

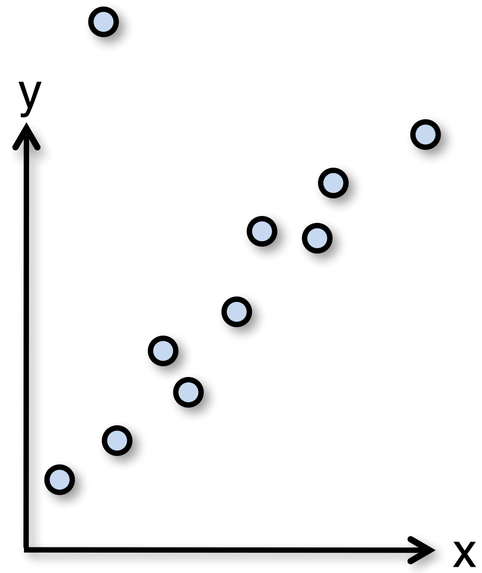


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC



- Back to linear regression
- How do we generate a hypothesis?

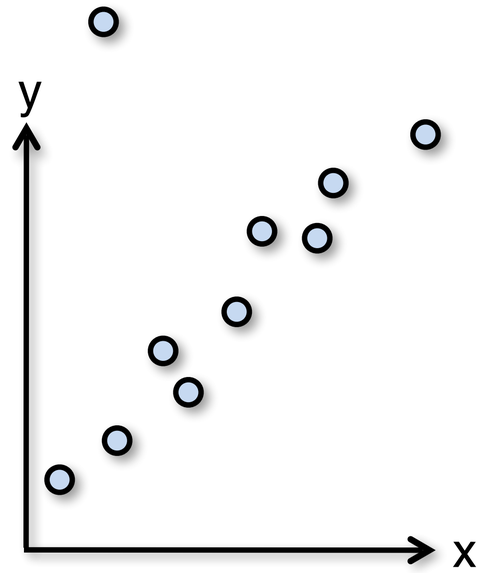


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC



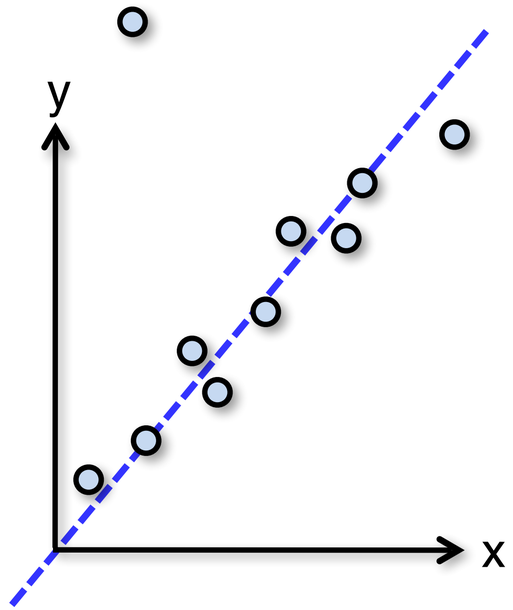
- Back to linear regression
- How do we generate a hypothesis?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- Back to linear regression
- How do we generate a hypothesis?



Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html