ADVANCES IN COMPUTER VISION

MODULE 1: Geometry

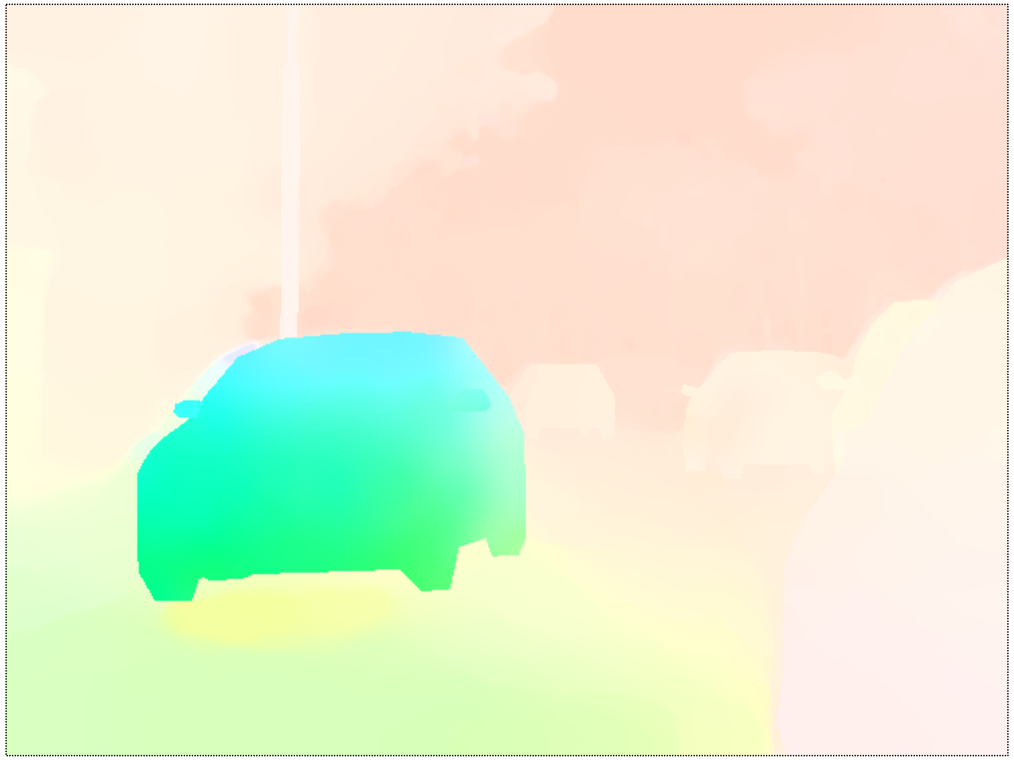
CORRESPONDENCE 2: POINT TRACKING, SCENE FLOW, FEATURE MATCHING

Prof. Vincent Sitzmann

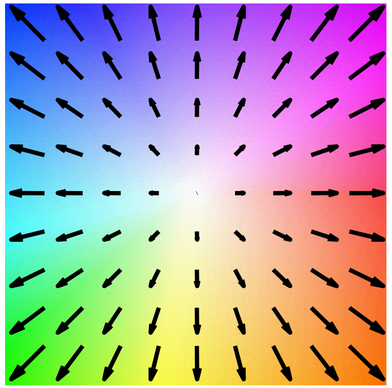
Last time on 6.8300

Optical flow: 2D motion of every pixel

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



Optical flow (2D motion vector)

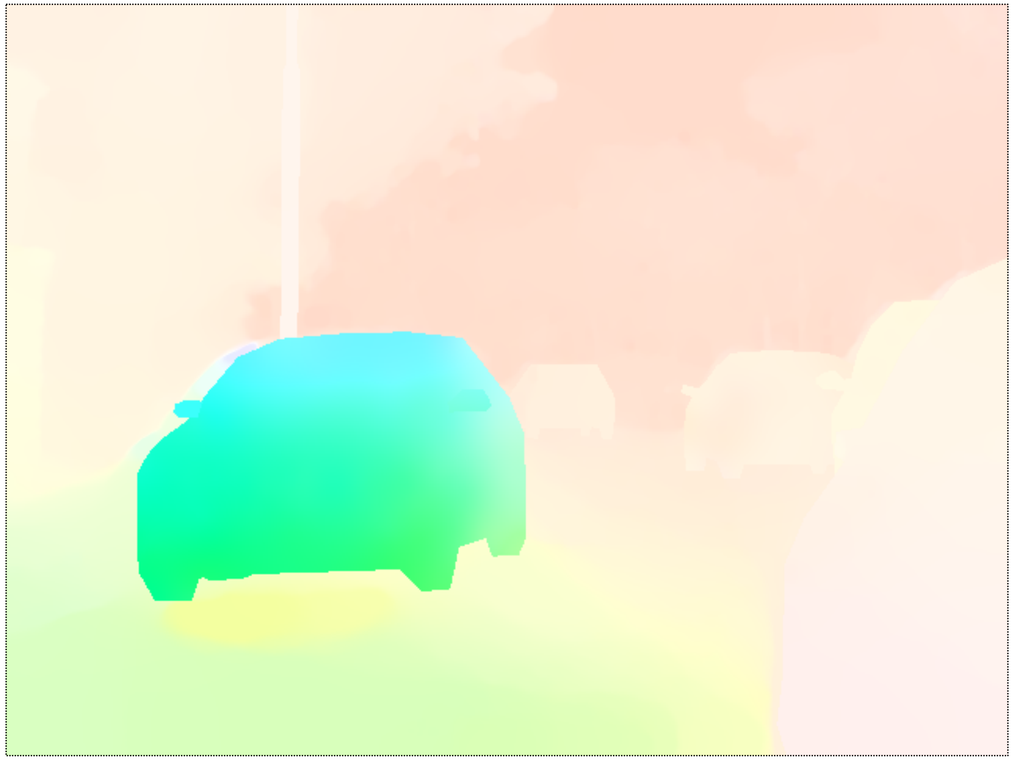


Color key
[Baker *et al.* IJCV'11]

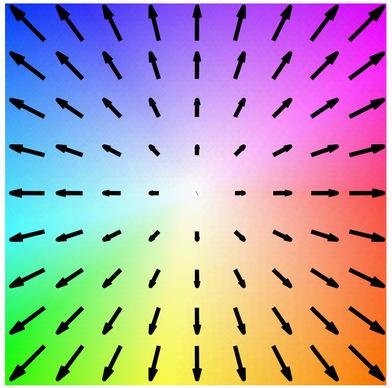
Slide credit: Deqing Sun

Optical flow: 2D motion of every pixel

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



Optical flow (2D motion vector)



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

Slide credit: Deqing Sun

Resolving ambiguities: Lucas-Kanade

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

Resolving ambiguities: Lucas-Kanade

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

Resolving ambiguities: Lucas-Kanade

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

Cost / Correspondence Volumes

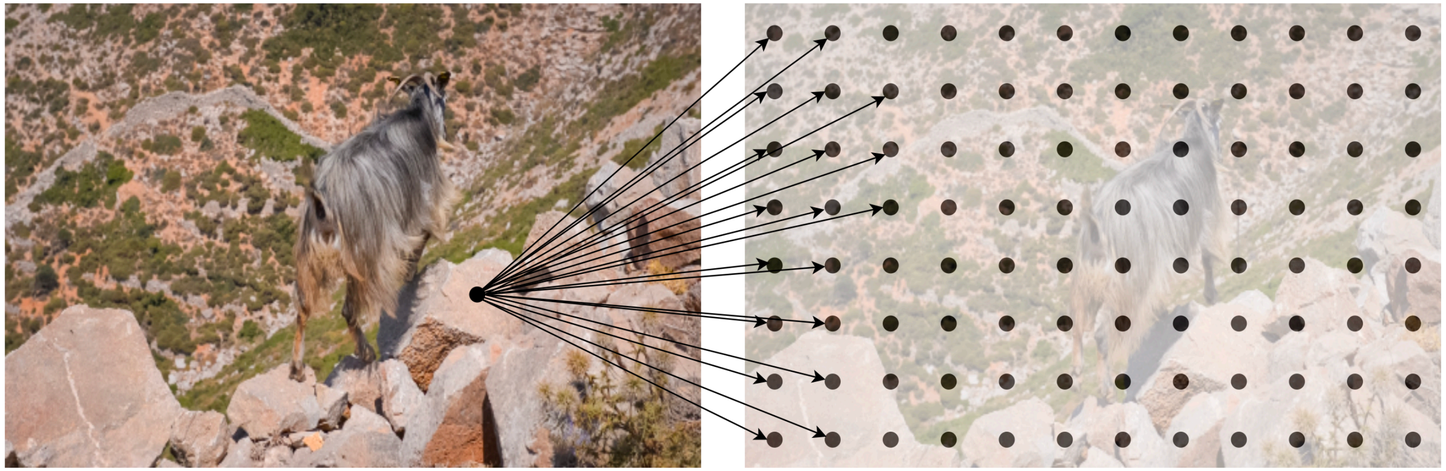
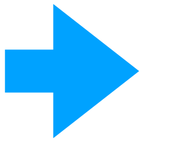
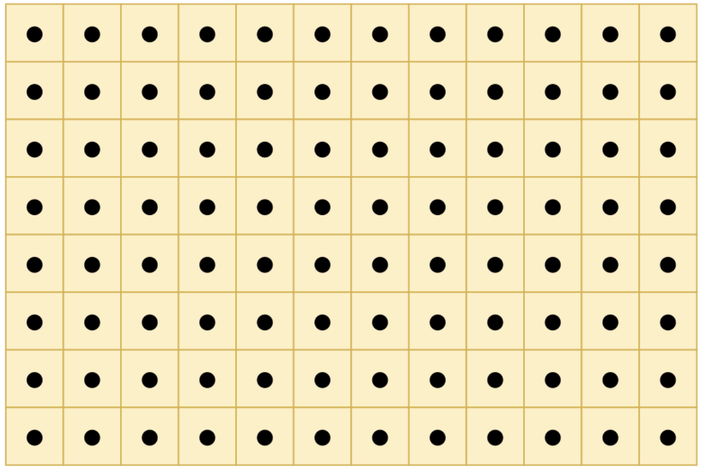


Image 1

Image 2

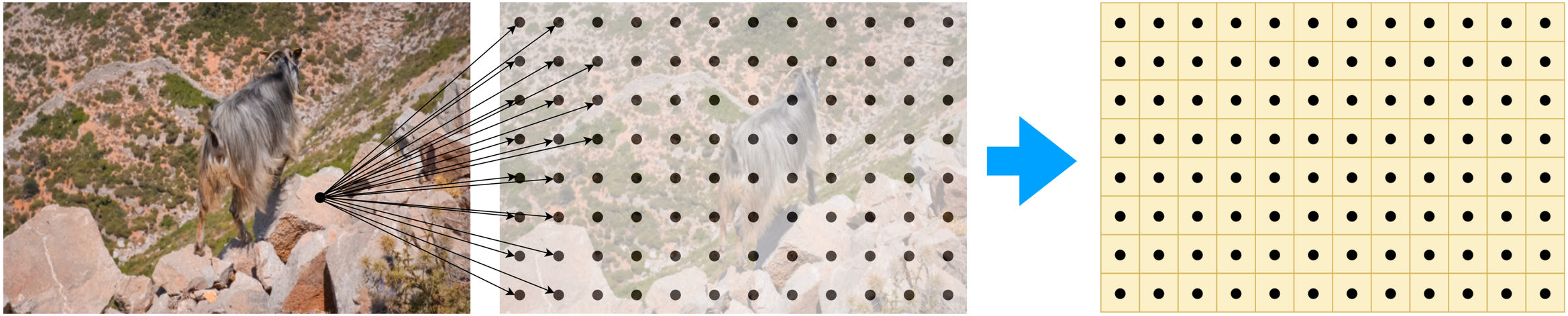




$\mathbf{C}^1 \in H \times W \times H \times W$

Image Courtesy: RAFT. *Teed et. al.*

Cost / Correspondence Volumes



Idea: Correlate every pixel / patch in image 1 with every pixel in image 2, yielding a 4D **correlation volume.**

Image Courtesy: RAFT. *Teed et. al.*

Similarity between patches (cost volume)

Image 1

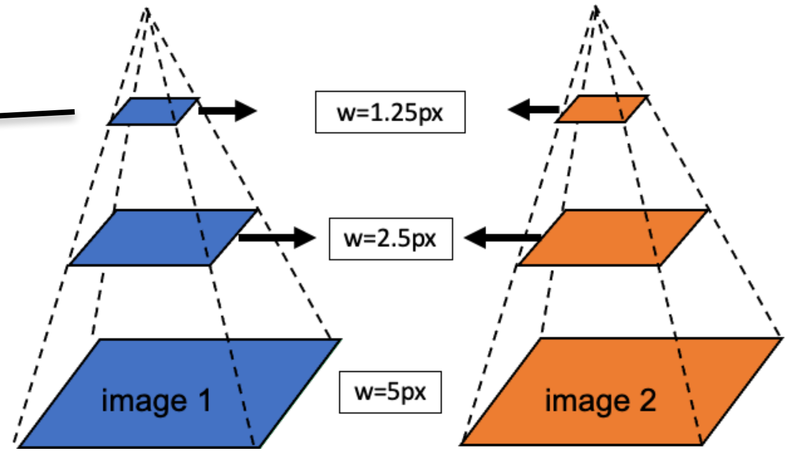
Cost volume (darker, more similar)

Coarse-to-fine iterative estimation

Current estimate

$$I_{t+1}\left(\mathbf{q}+\mathbf{w_p}\right) = I_{t+1}\left(\mathbf{q}+\mathbf{w_p}^{k}+\delta\mathbf{w_p}\right) = \mathbf{I_w}(\mathbf{q}+\delta\mathbf{w_p})$$

Small increment

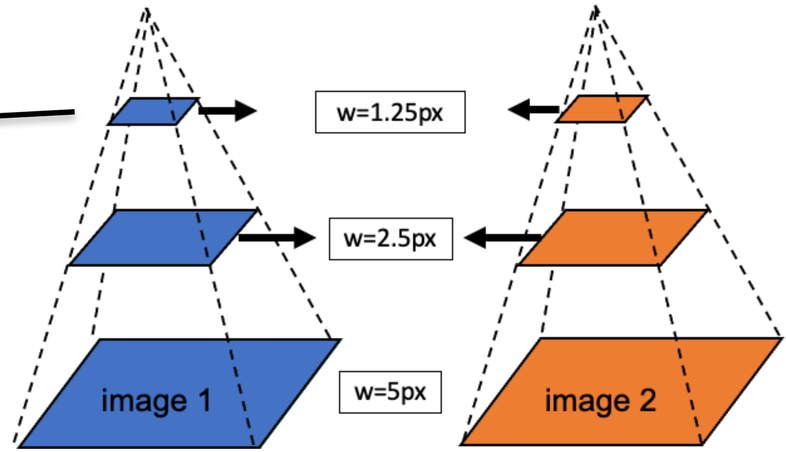


Coarse-to-fine iterative estimation

Current estimate

$$I_{t+1}\left(\mathbf{q}+\mathbf{w_p}\right) = I_{t+1}\left(\mathbf{q}+\mathbf{w_p}^{k}+\delta\mathbf{w_p}\right) = \mathbf{I_w}(\mathbf{q}+\delta\mathbf{w_p})$$

Small increment



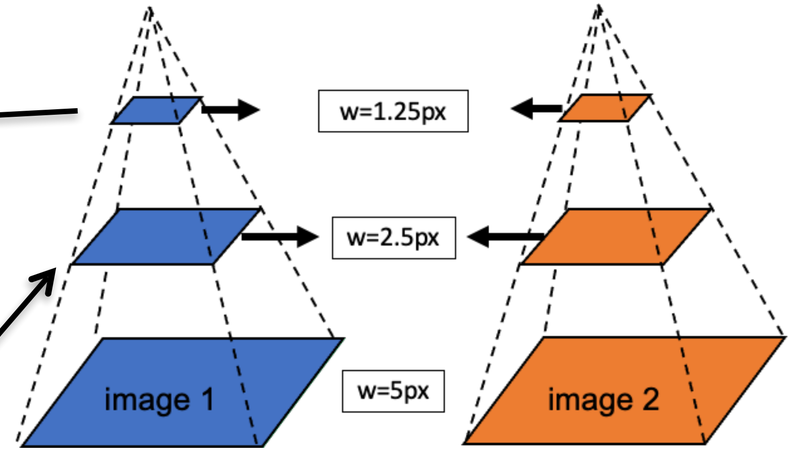
Warped image:

Coarse-to-fine iterative estimation

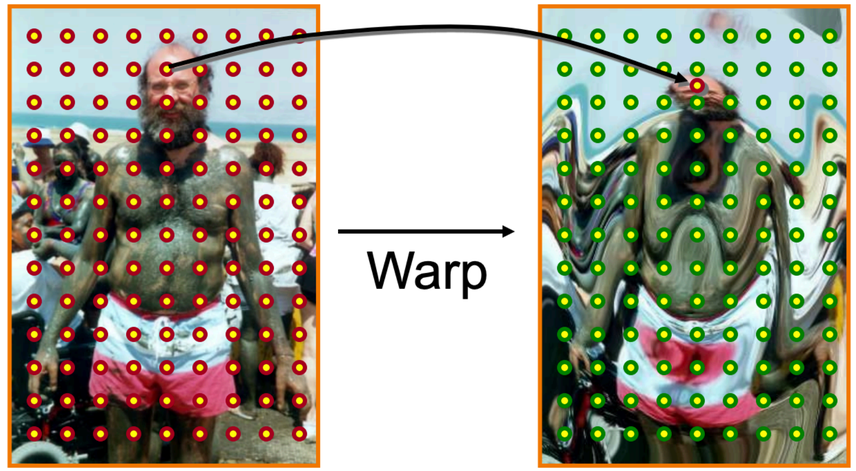
Current estimate

$$I_{t+1}\left(\mathbf{q}+\mathbf{w}_{\mathbf{p}}\right)=I_{t+1}\left(\mathbf{q}+\mathbf{w}_{\mathbf{p}}{}^{k}+\delta\mathbf{w}_{\mathbf{p}}\right)=\mathbf{I}_{\mathbf{w}}(\mathbf{q}+\delta\mathbf{w}_{\mathbf{p}})$$

Small increment



Warped image: $I_{\mathrm{w}}(\mathbf{q})=I_{t+1}\left(\mathbf{q}+\mathbf{w}_{\mathbf{p}}^{k}\right)$

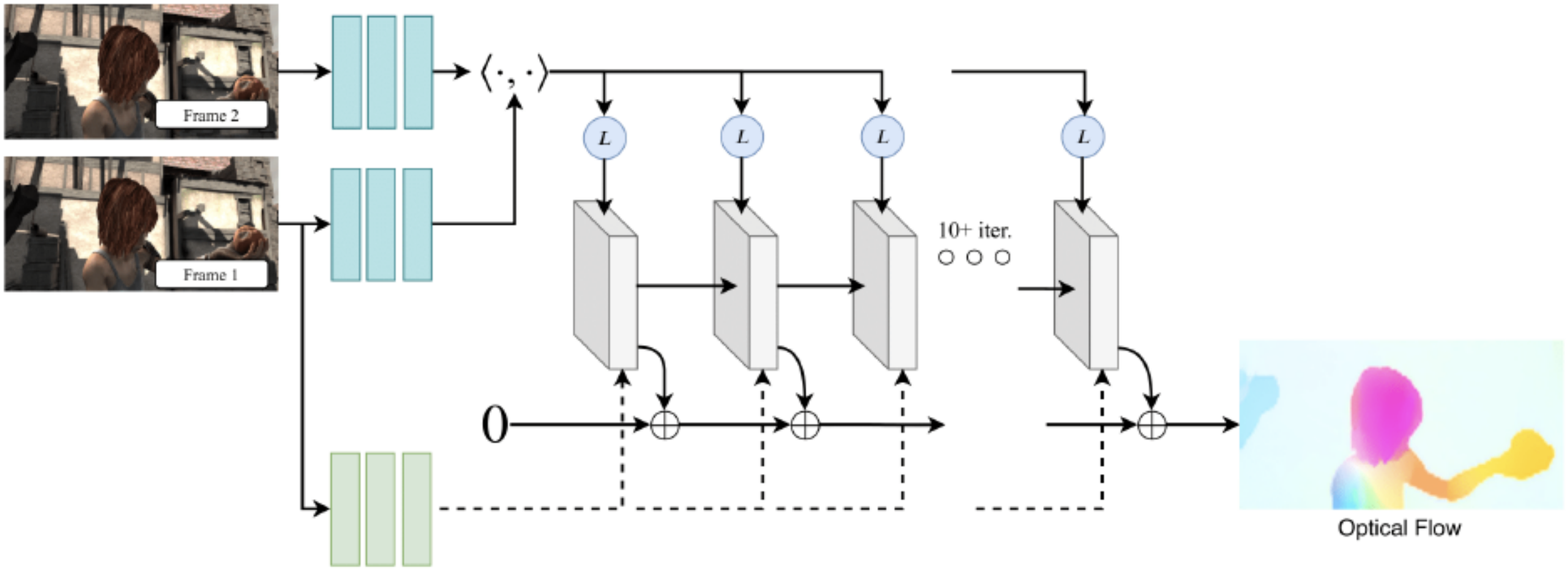


Source image

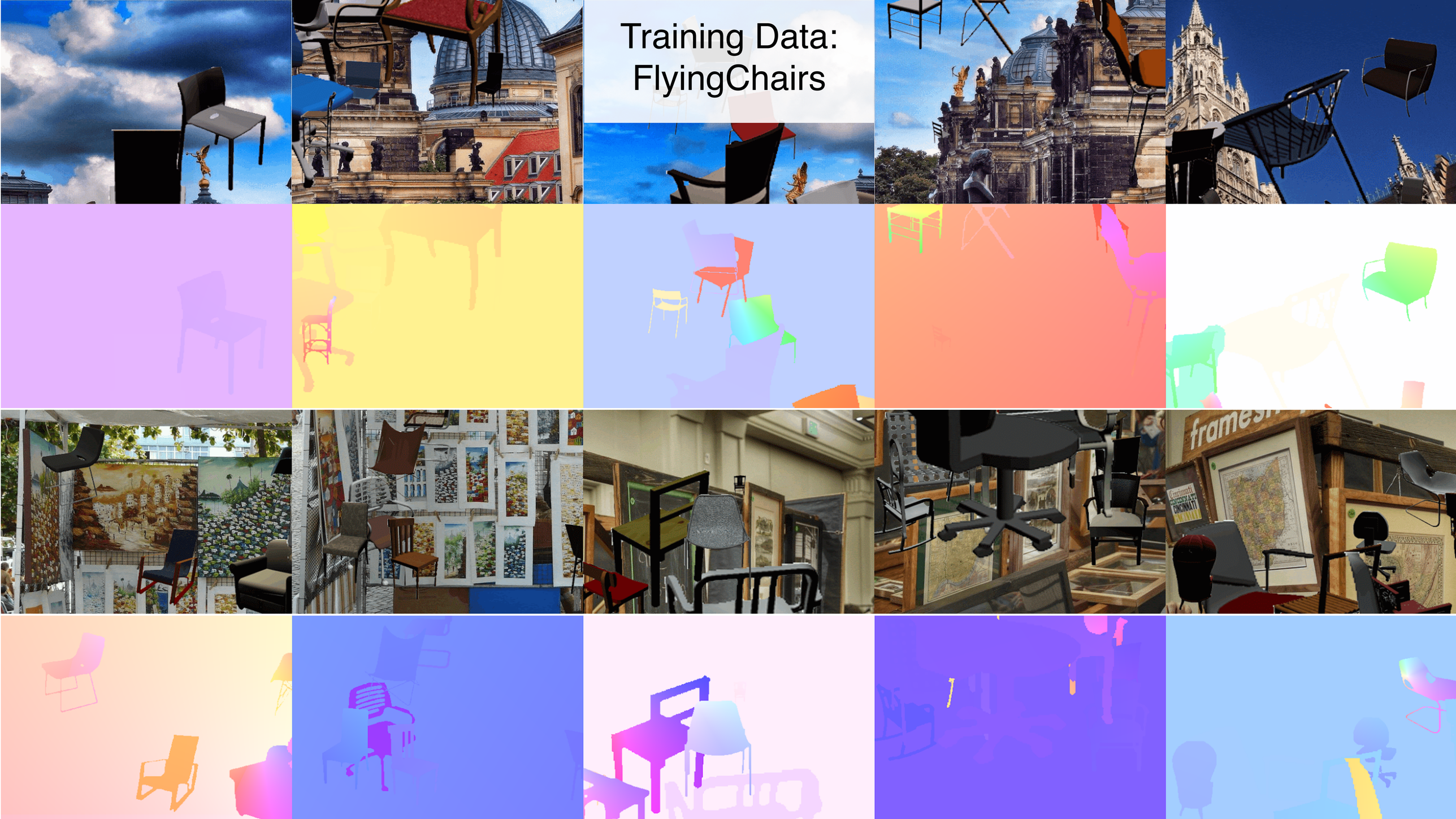
Destination image

RAFT: Recurrent All-pairs Field Transforms

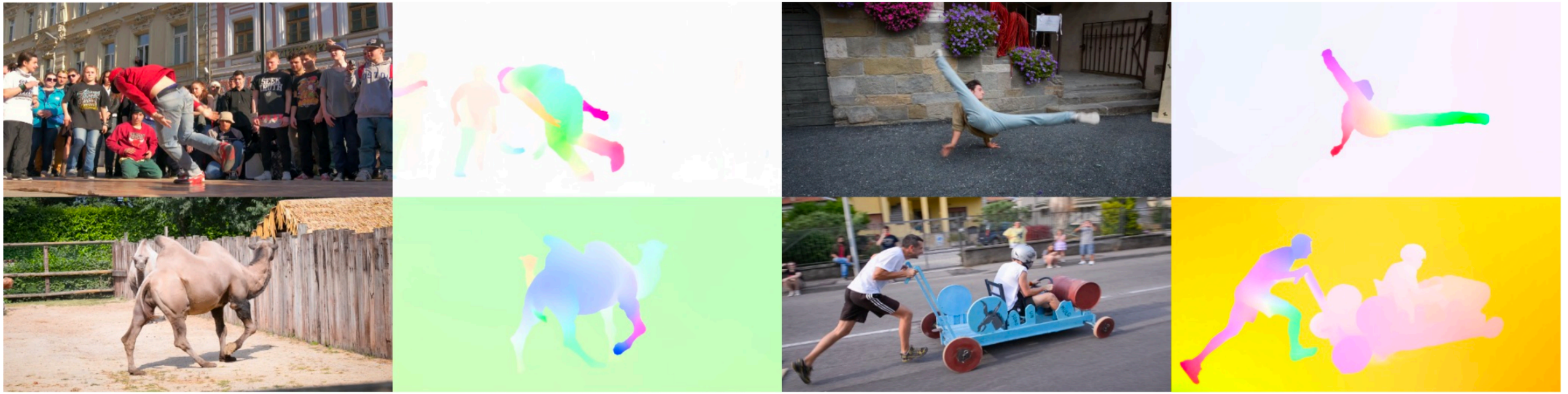
[Teed and Deng ECCV 2020 Best paper]



Slide credit: Teed and Deng


Training Data:
FlyingChairs

Visual results on Davis (real-world)



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

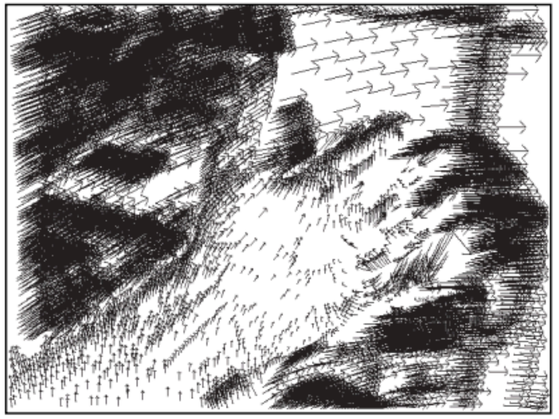
This paper describes a new approach to motion estimation in video. We represent video motion using a set of particles. Each particle is an image point sample with a long-duration trajectory and other properties. To optimize these particles, we measure point-based matching along the particle trajectories 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-

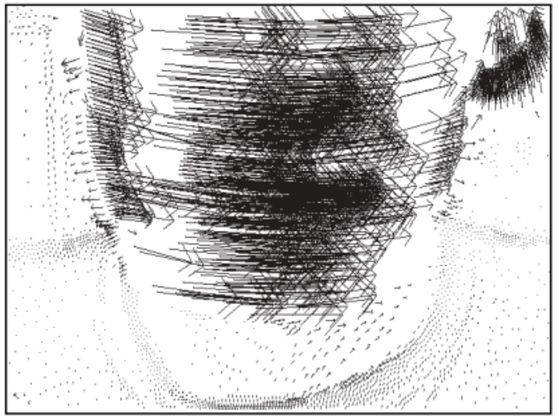
Video 1, Frame 0



Particle Correspondences

Video 1, Frame 25

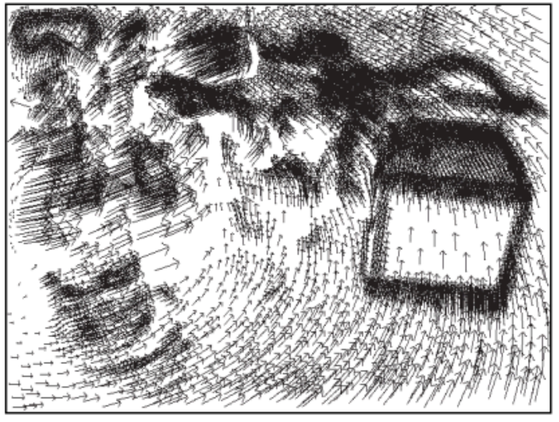
Video 2, Frame 0



Particle Correspondences

Video 2, Frame 25

Video 3, Frame 0



Particle Correspondences

Video 3, Frame 25

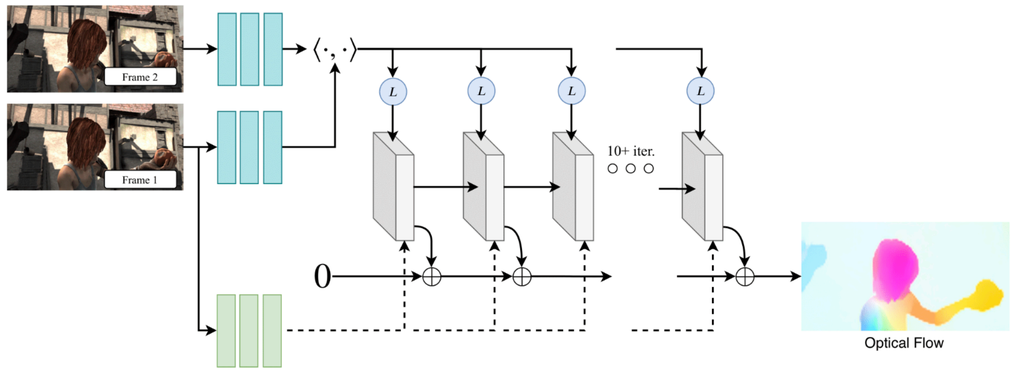
Two approaches to Motion Estimation

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

Two approaches to Motion Estimation

Point Tracking

Long-term tracking of individual points

Two approaches to Motion Estimation

Optical Flow

Dense correspondences between a pair of frames

Point Tracking

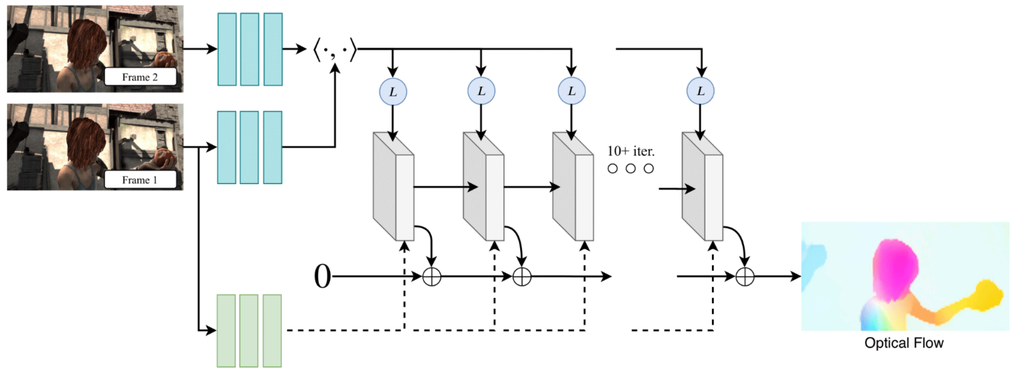
Long-term tracking of individual points

Two approaches to Motion Estimation

Optical Flow

Dense correspondences between a pair of frames

RAFT

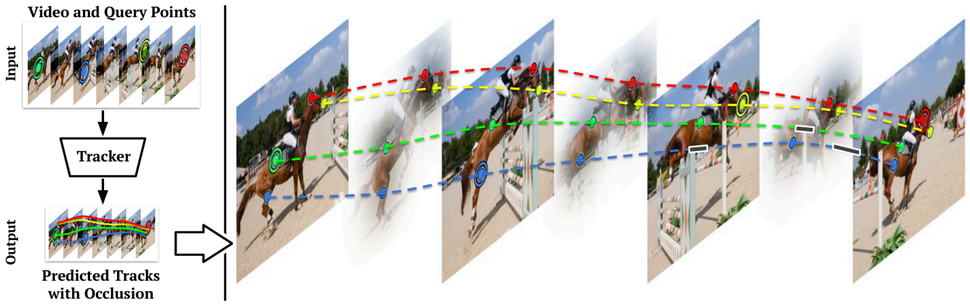


Point Tracking

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

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.

Particle Video Revisited

Initialize positions and features

(x_1, y_1)

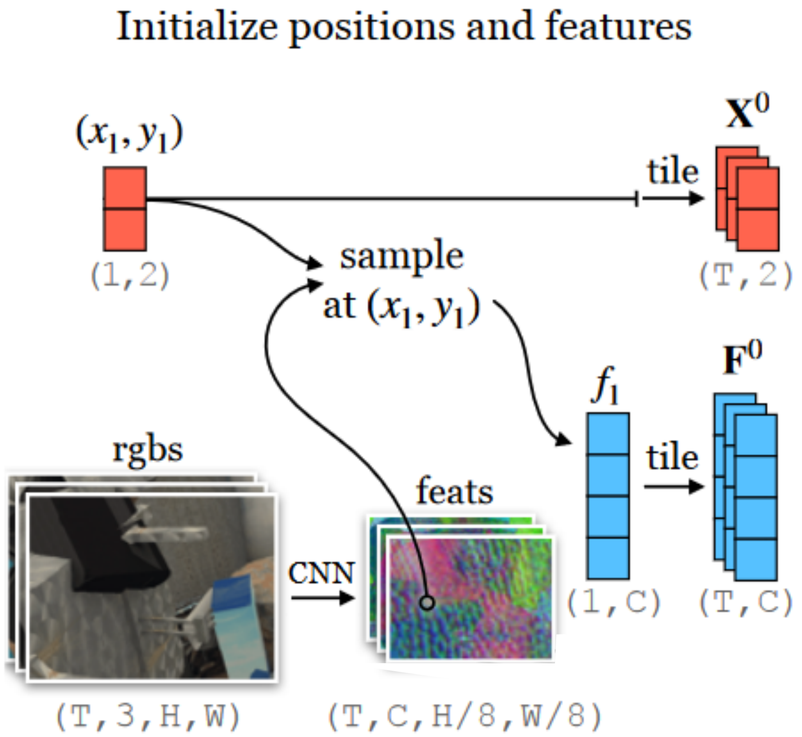
(1,2)

rgbs

(T,3,H,W)

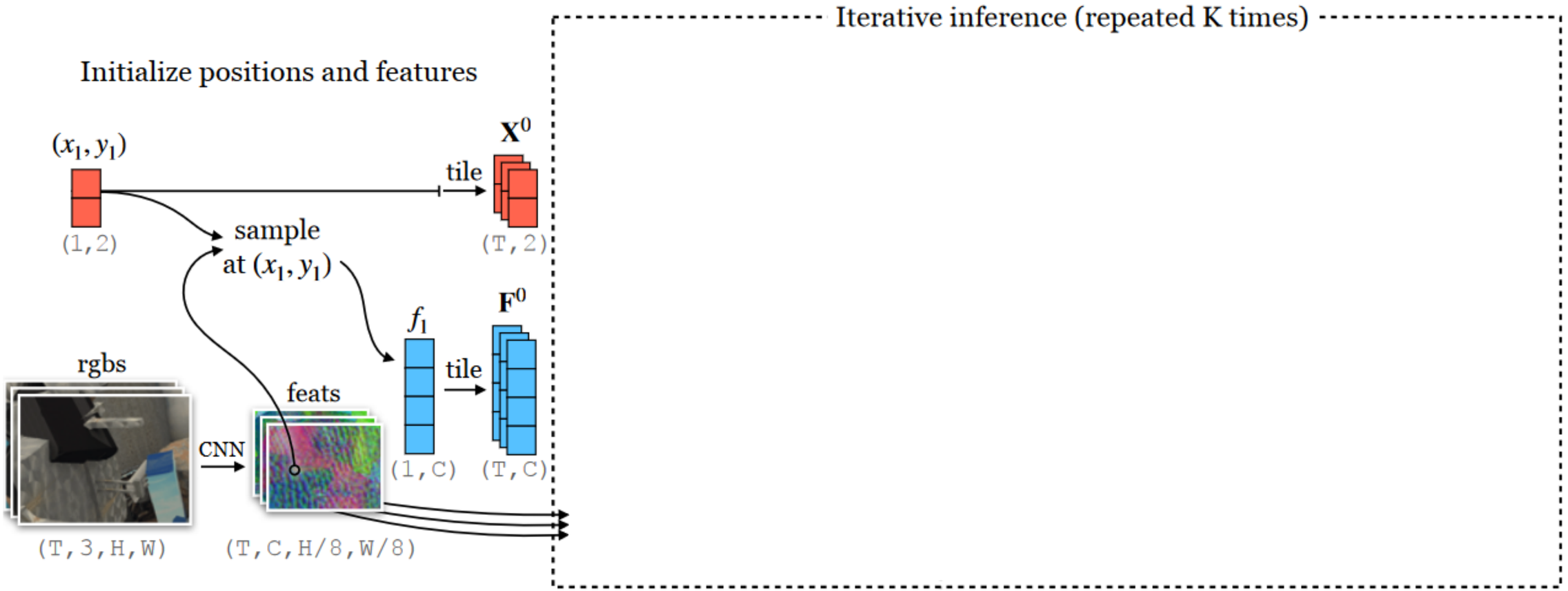
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



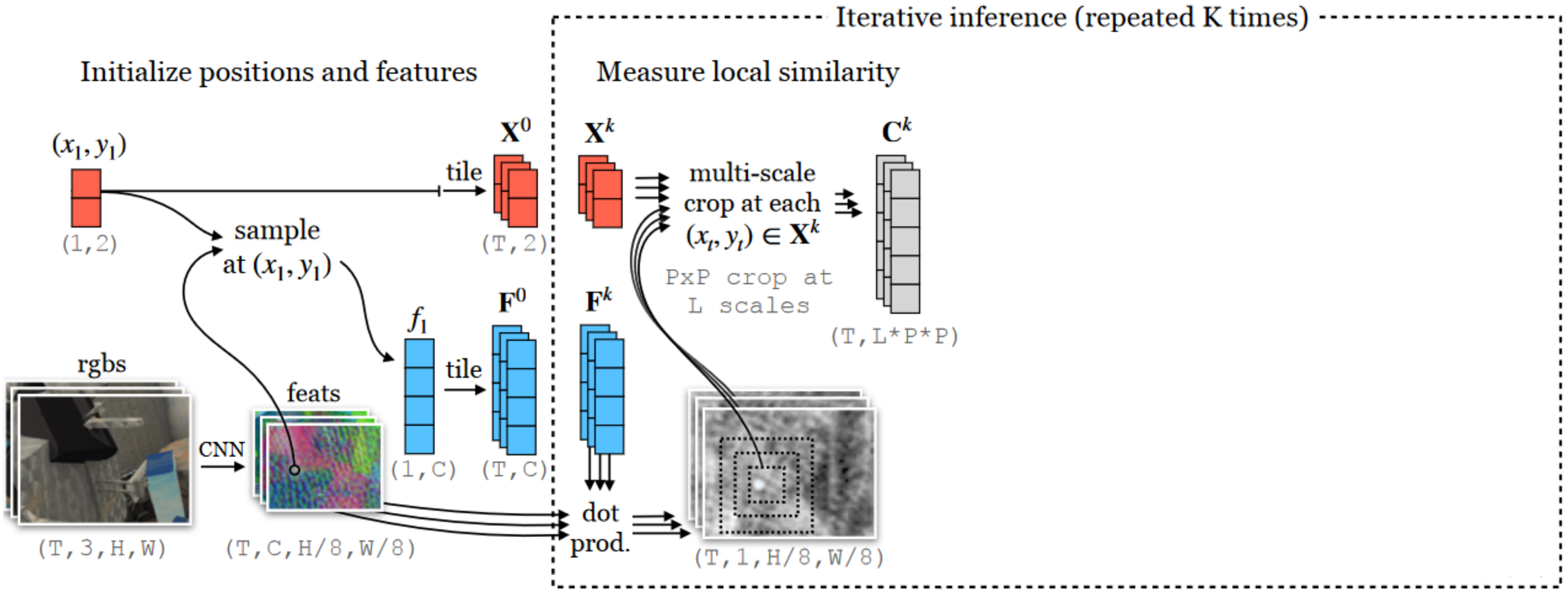
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



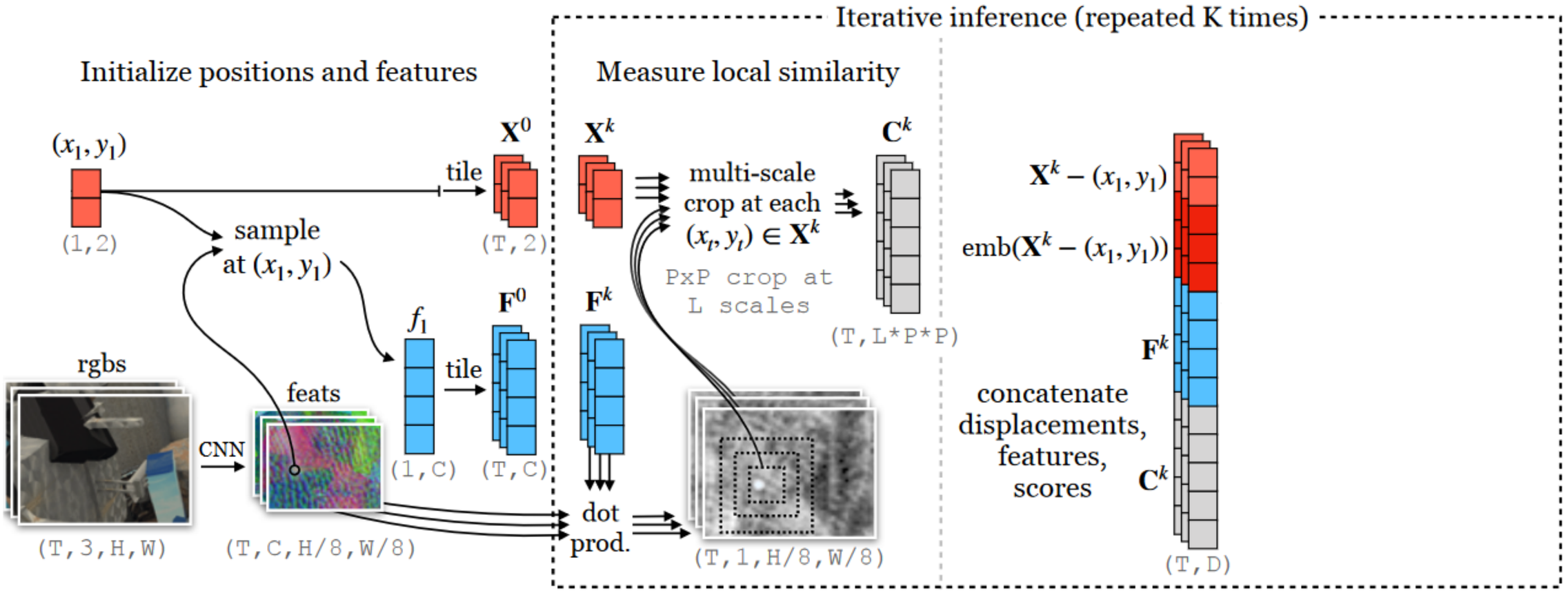
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



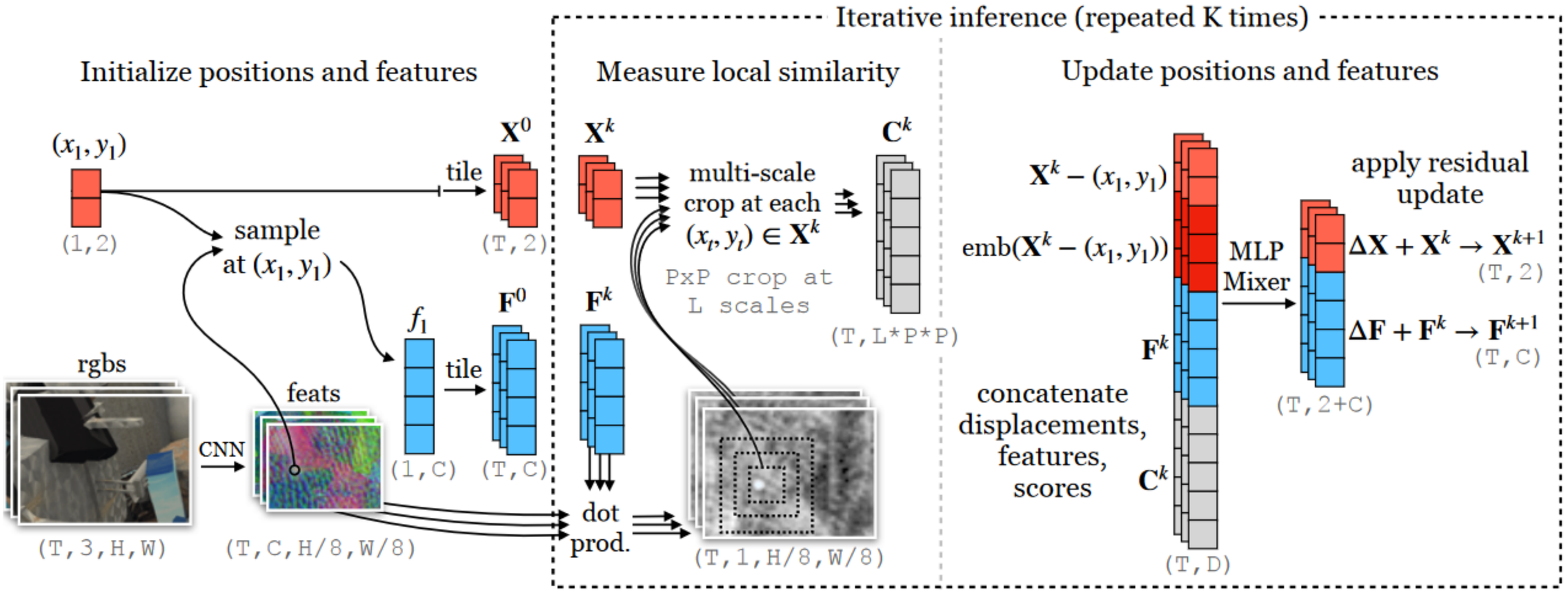
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



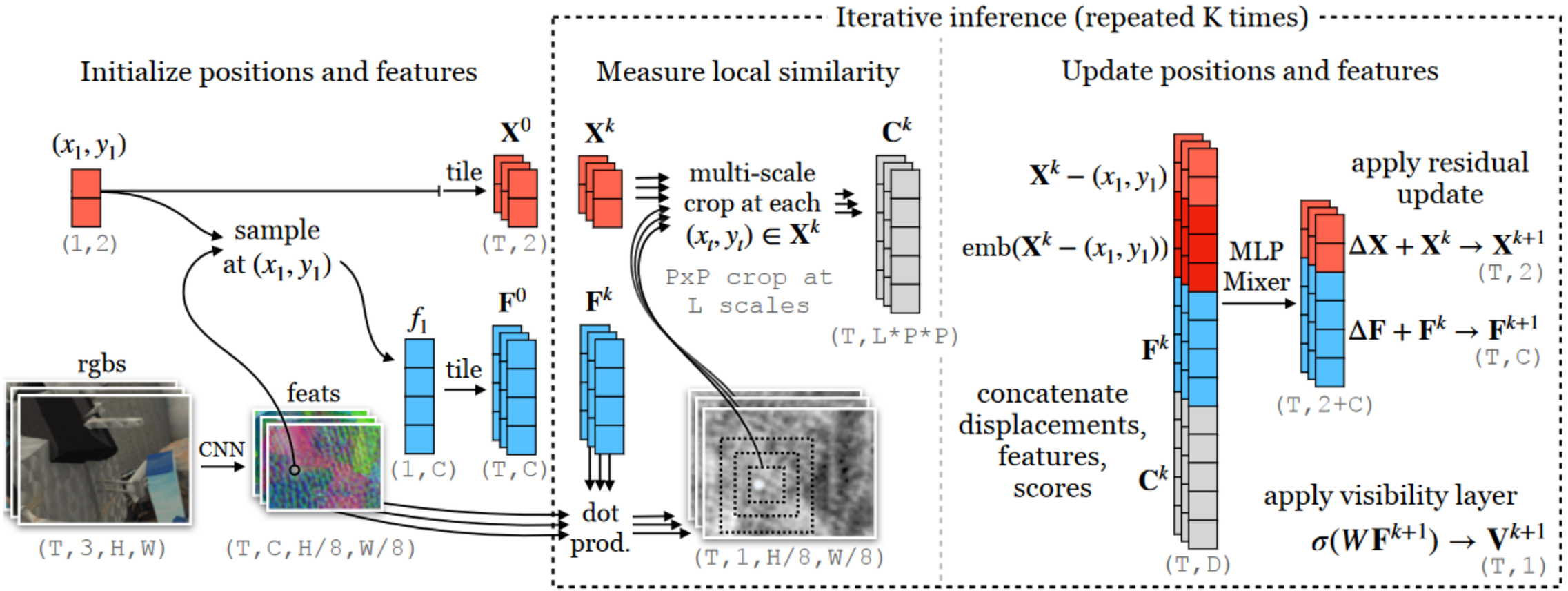
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



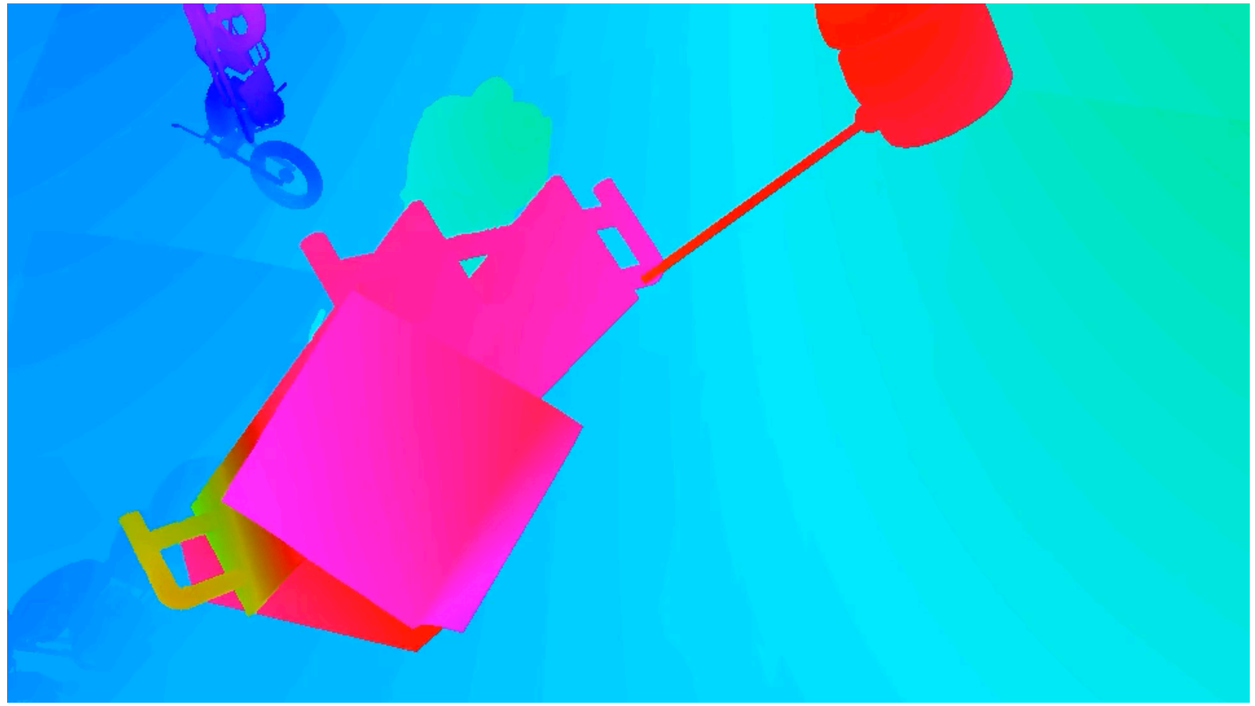
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited

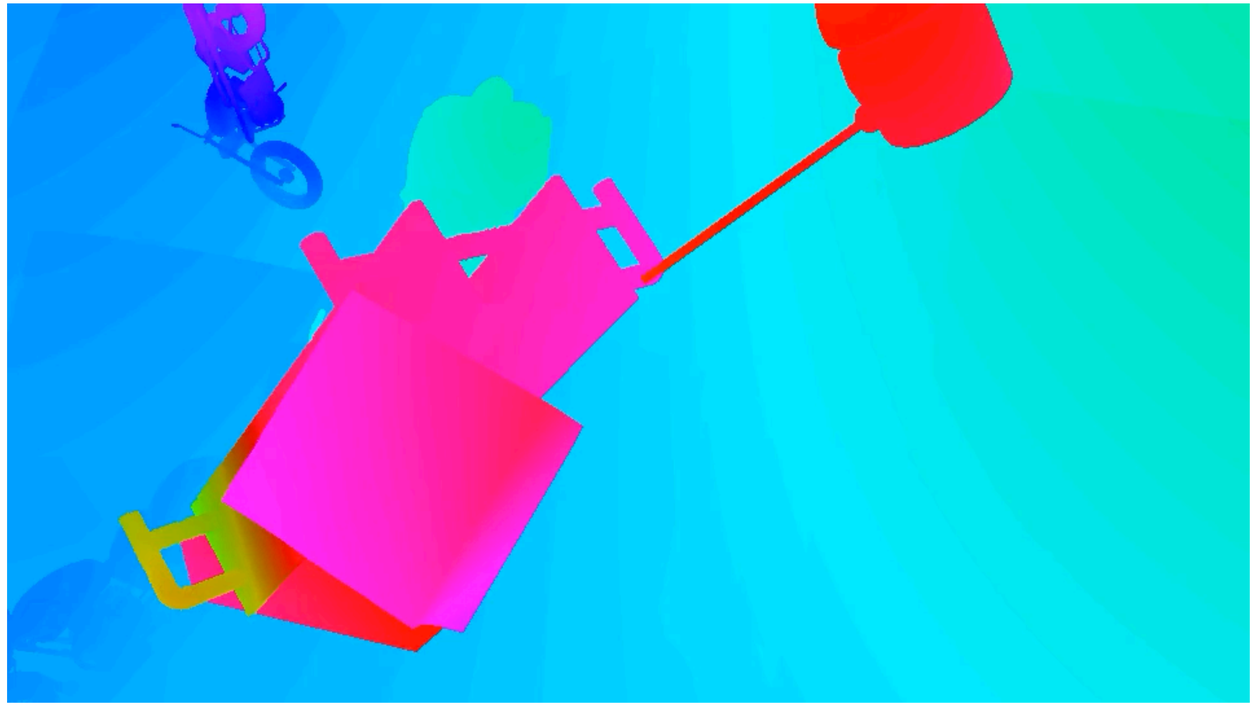


Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

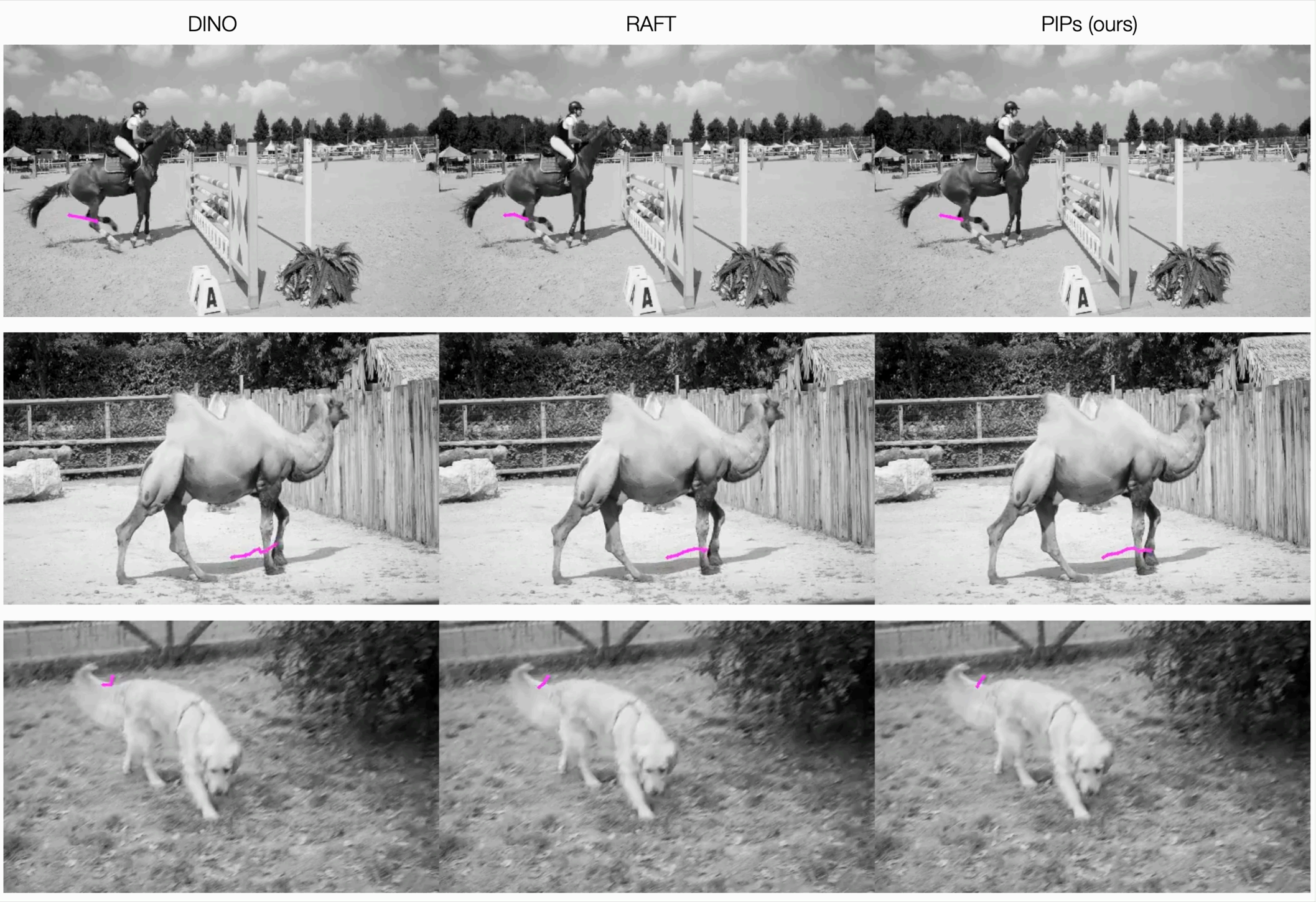
Trained Fully Supervised on Synthetic Data

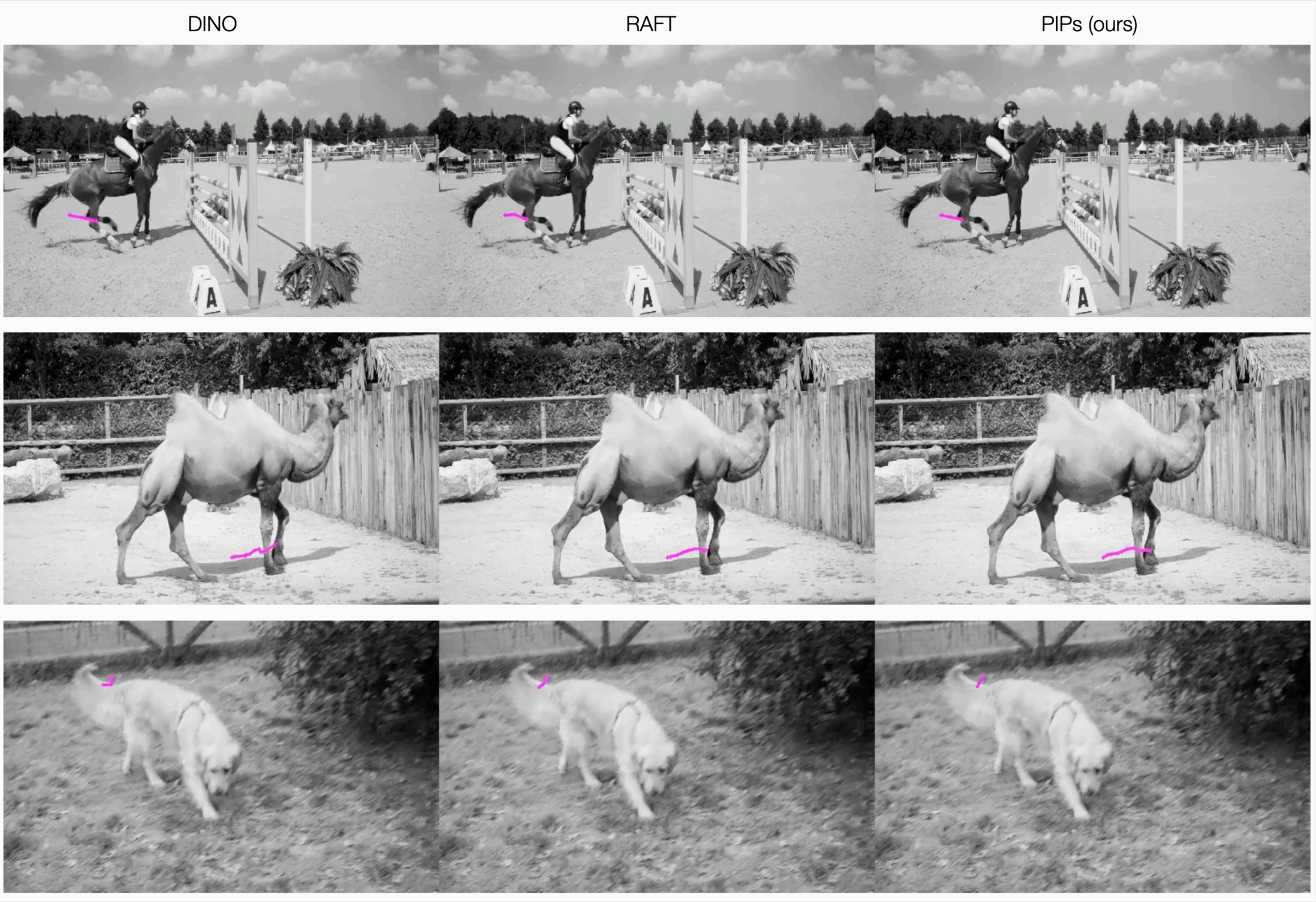


Trained Fully Supervised on Synthetic Data

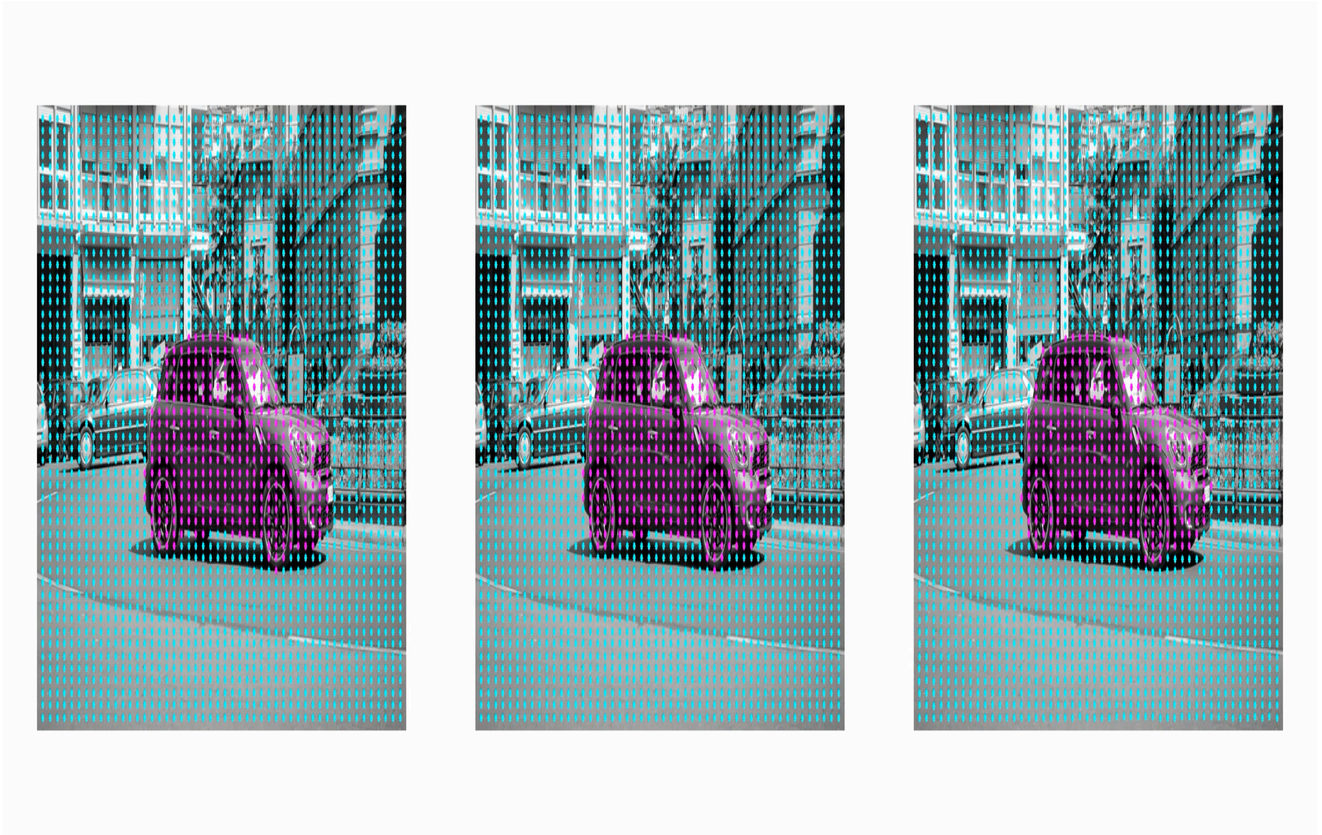


A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation, Mayer et al., 2016


DINO
RAFT
PIPs (ours)

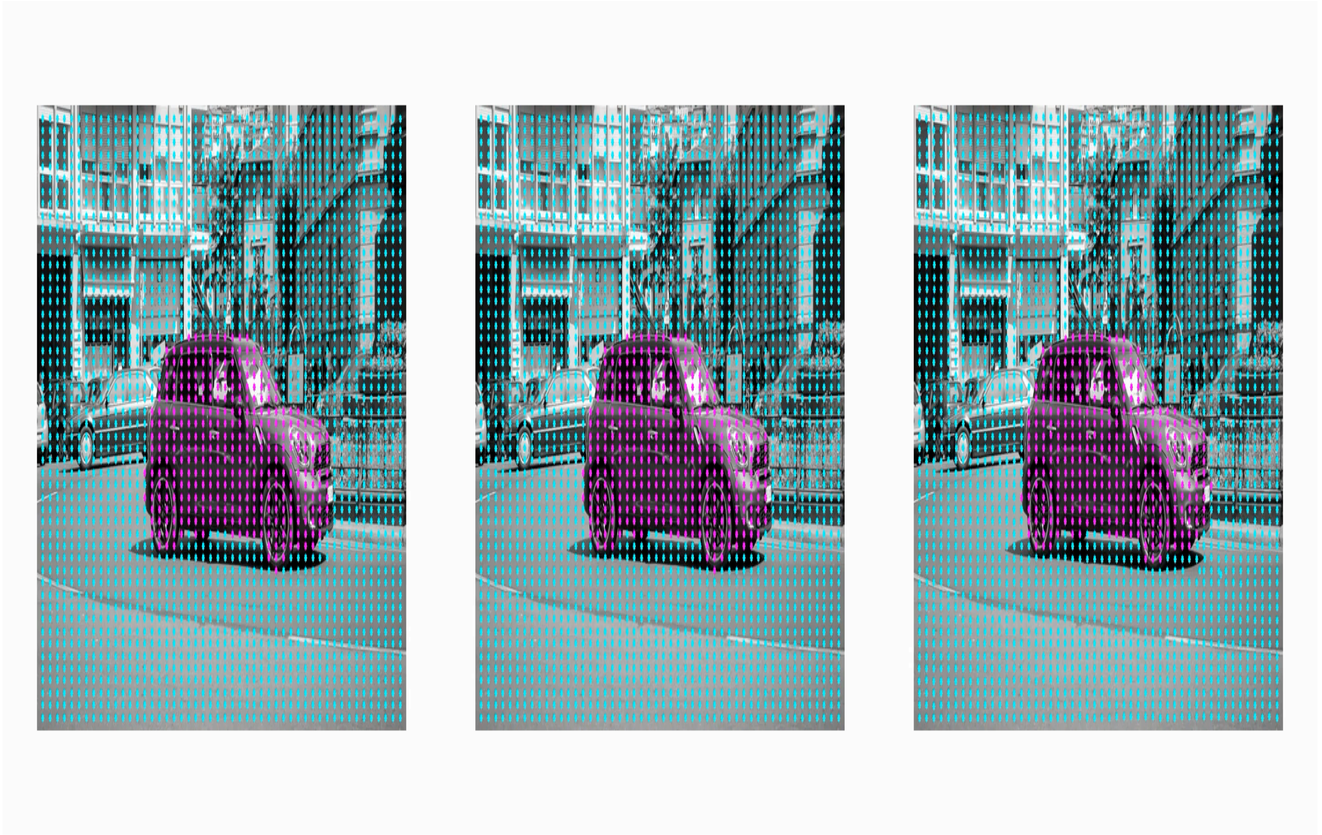

DINO
RAFT
PIPs (ours)

Single Point Tracking



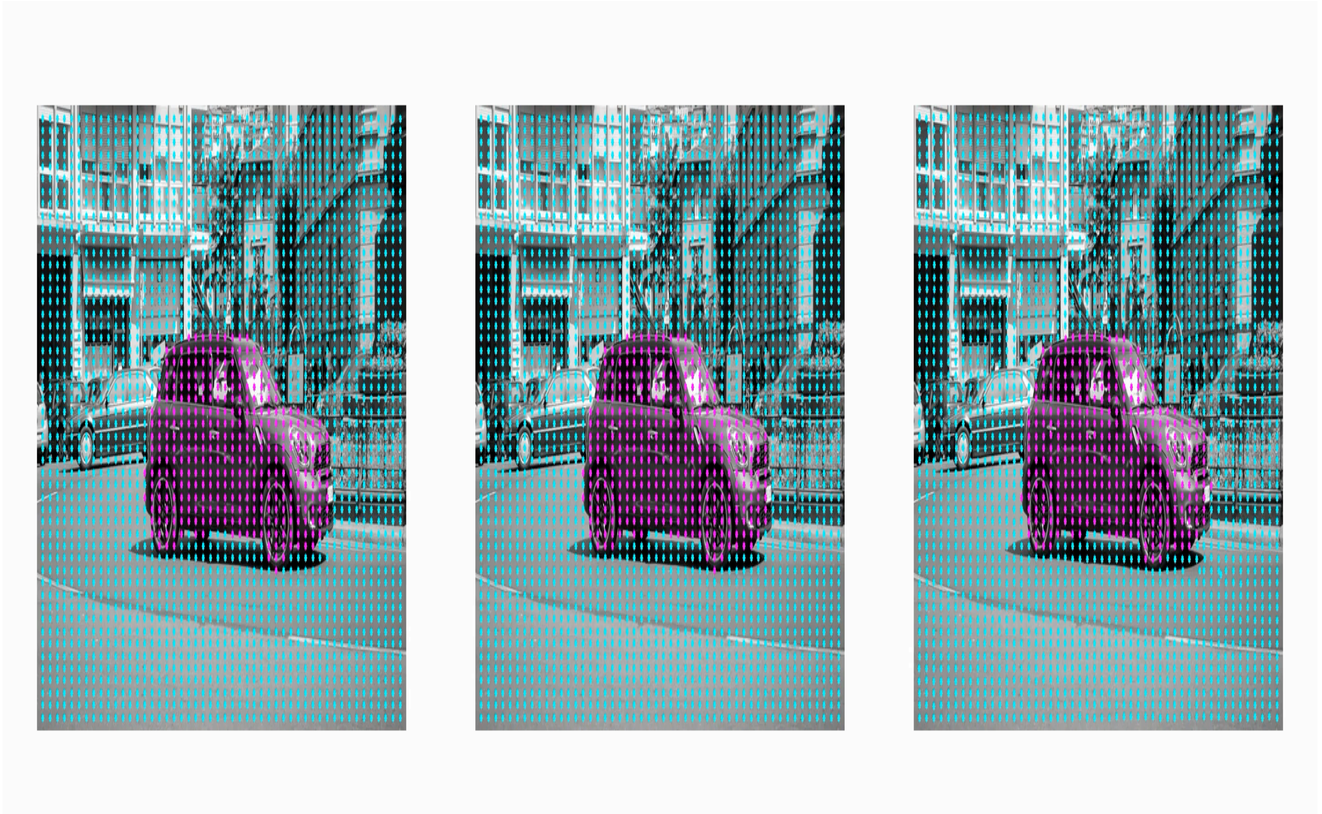
- Background points
- Object points

Single Point Tracking



- Background points
- Object points

Single Point Tracking

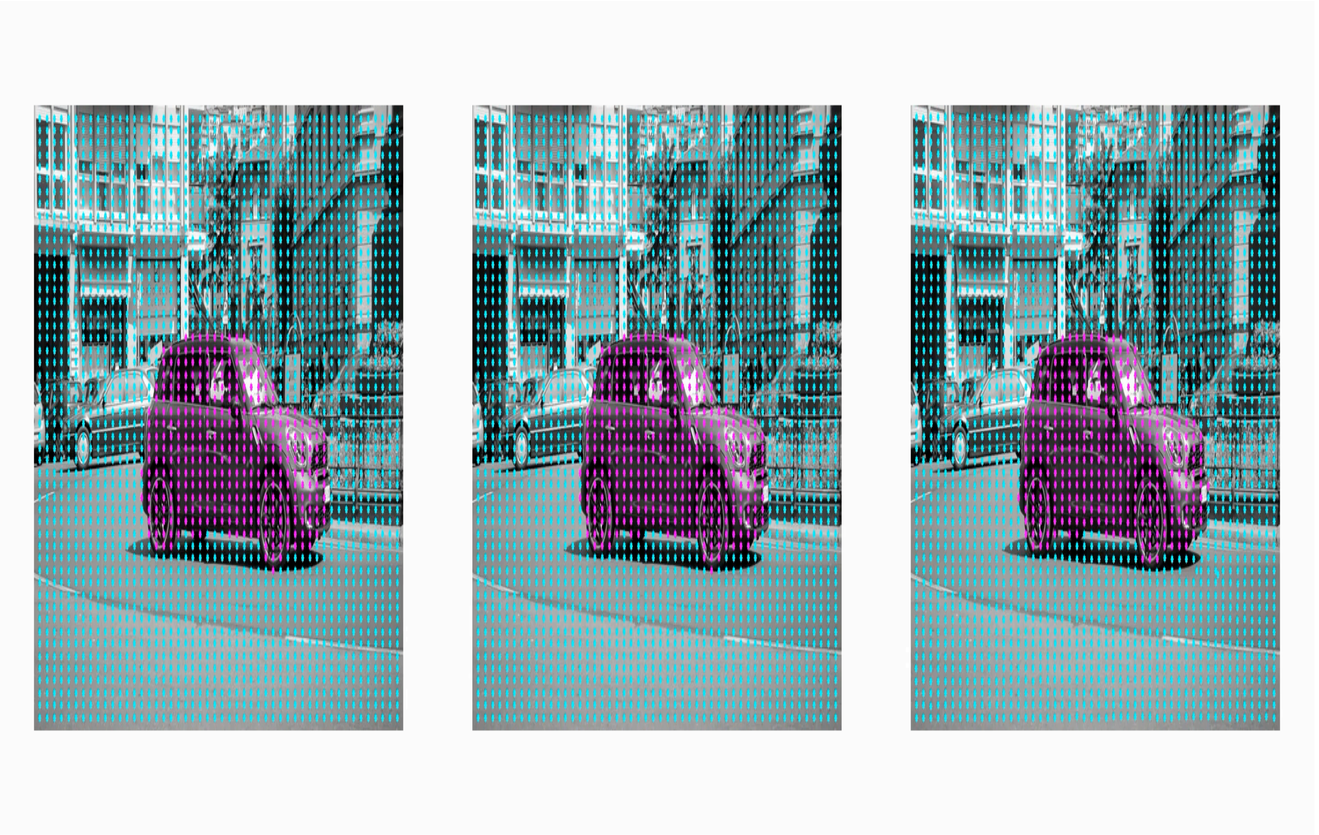


- Background points
- Object points

Tracking single points lacks global consistency

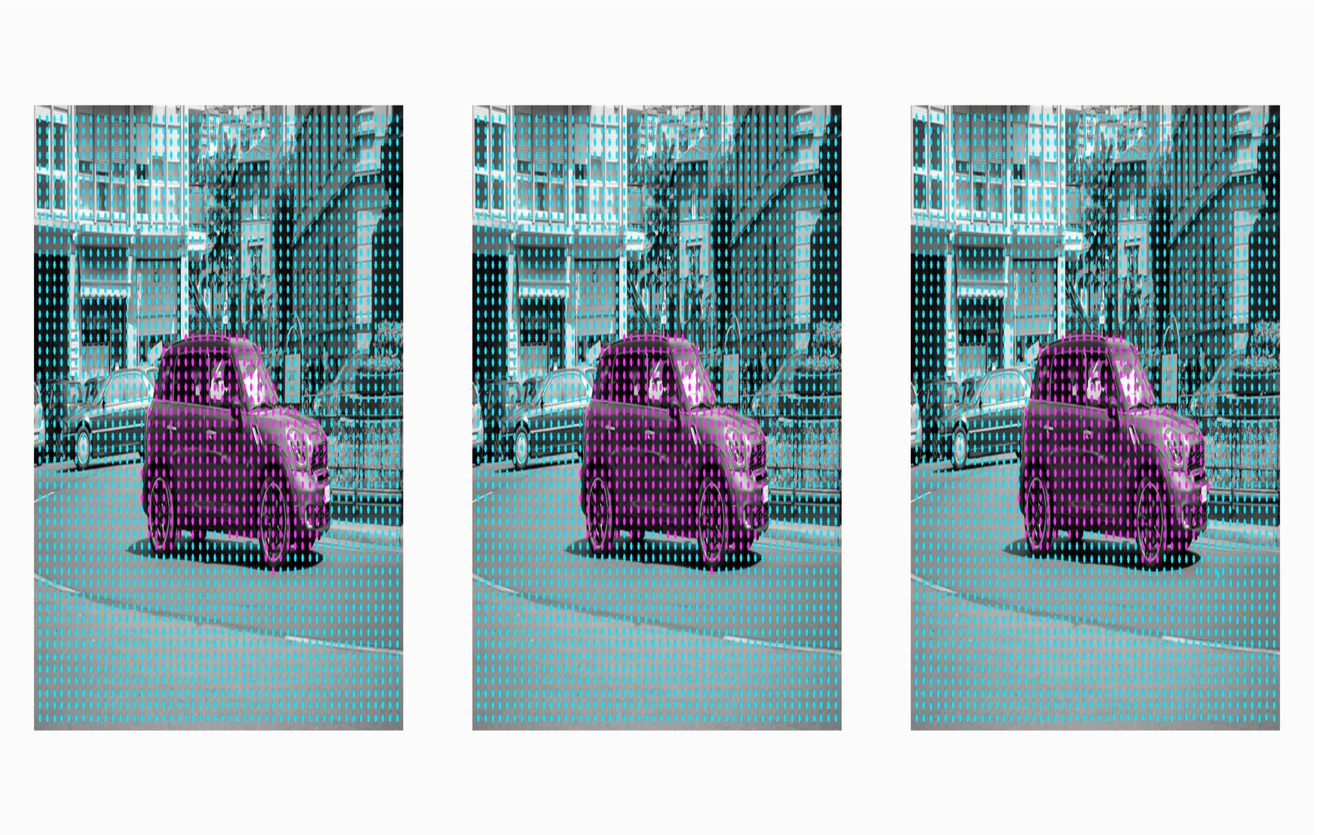
Points drift relative to each other

Tracking with Optical Flow



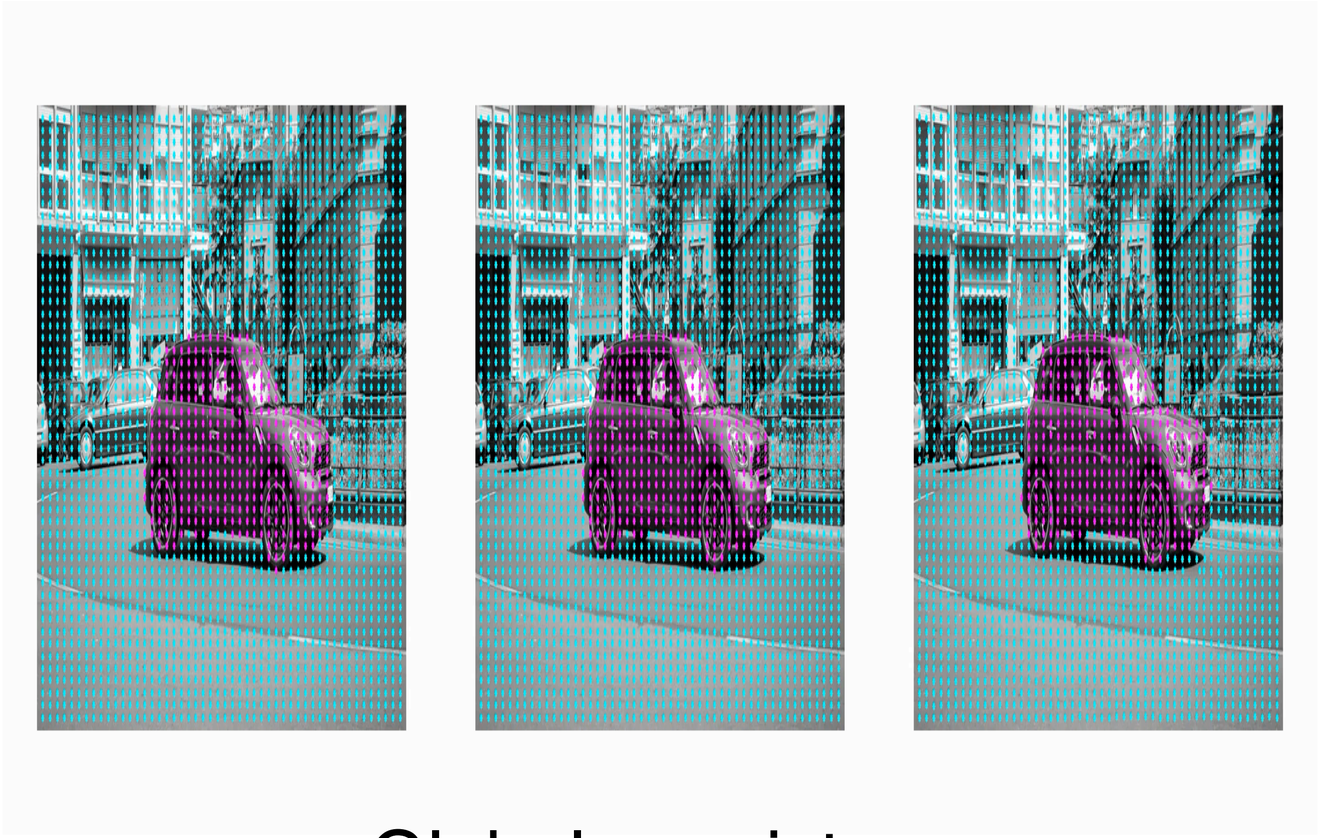
- Background points
- Object points

Tracking with Optical Flow



- Background points
- Object points

Tracking with Optical Flow

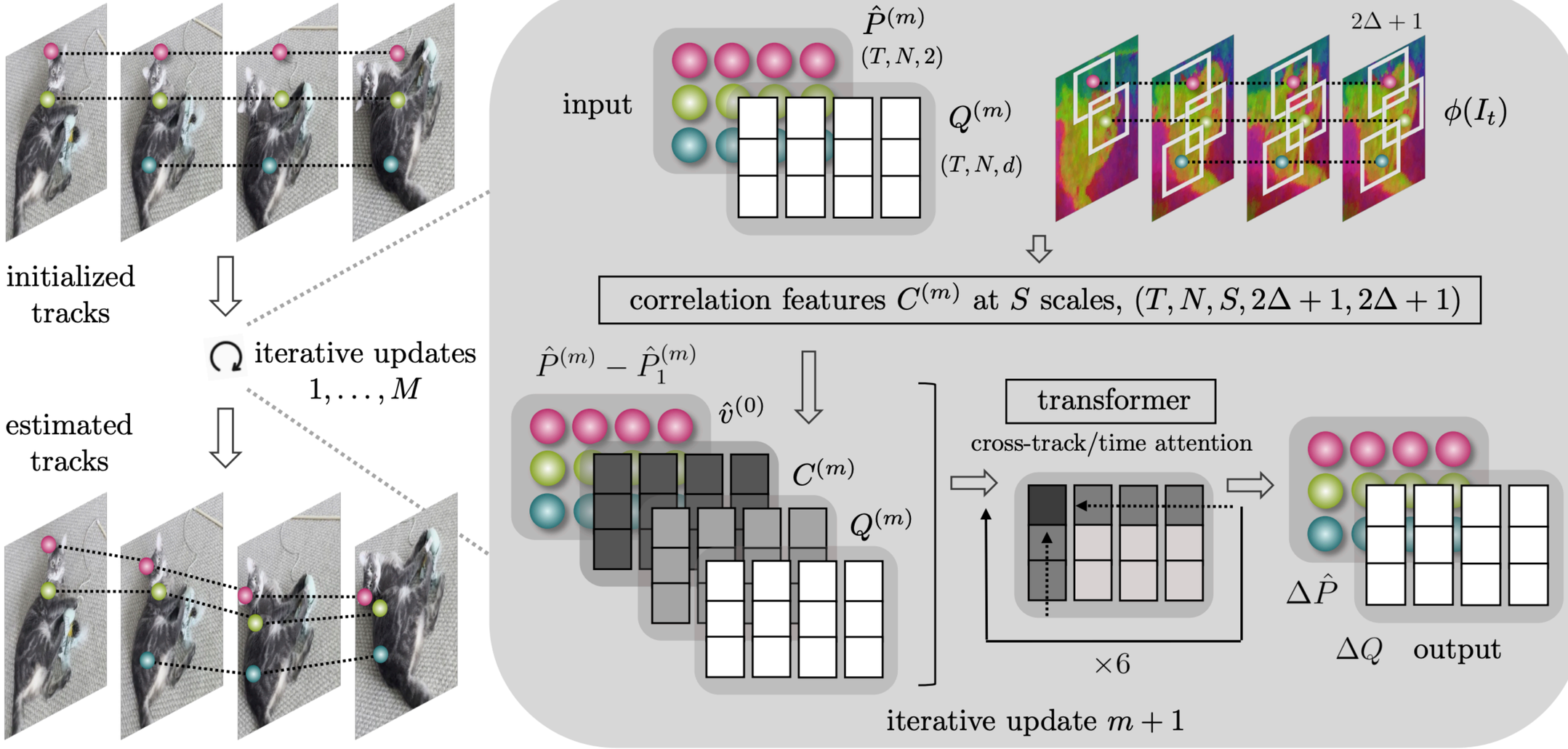


- Background points
- Object points

Global consistency

Occlusion problem: background accumulates on the object

CoTracker: It is Better to Track Together



Tracking Points Together

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

Tracking Points Together

- Global consistency from tracking many points

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

Tracking Points Together

- Global consistency from tracking many points
- Occlusion handling from long-term tracking

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

Tracking Points Together

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

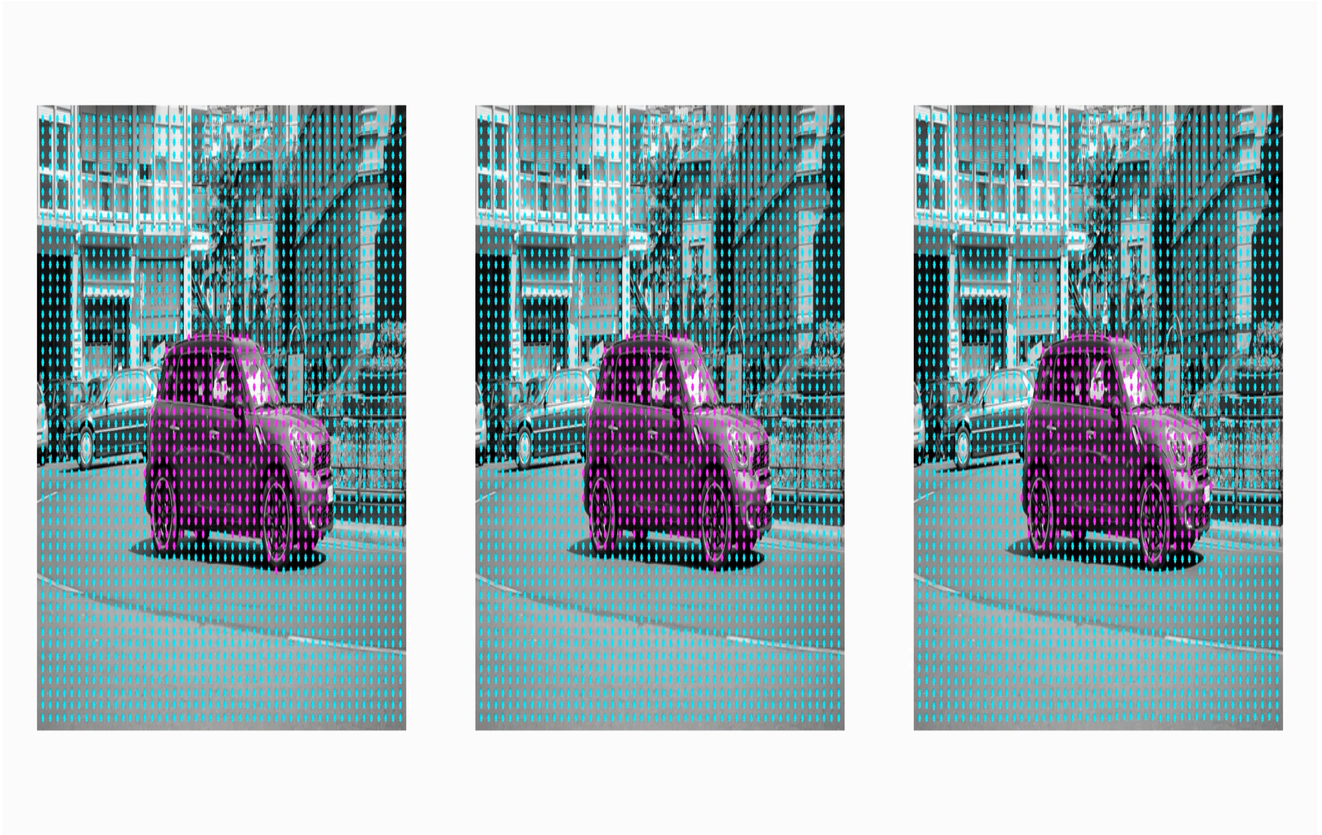
CoTracker: It is Better to Track Together, N Karaev et al., 2023

Tracking Points Together

- Global consistency from tracking many points
- Occlusion handling from long-term tracking
- Train on synthetic data only
- Main mechanism:
  - Transformer architecture
  - Factorised attention: space, time, and group (across tracks)

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

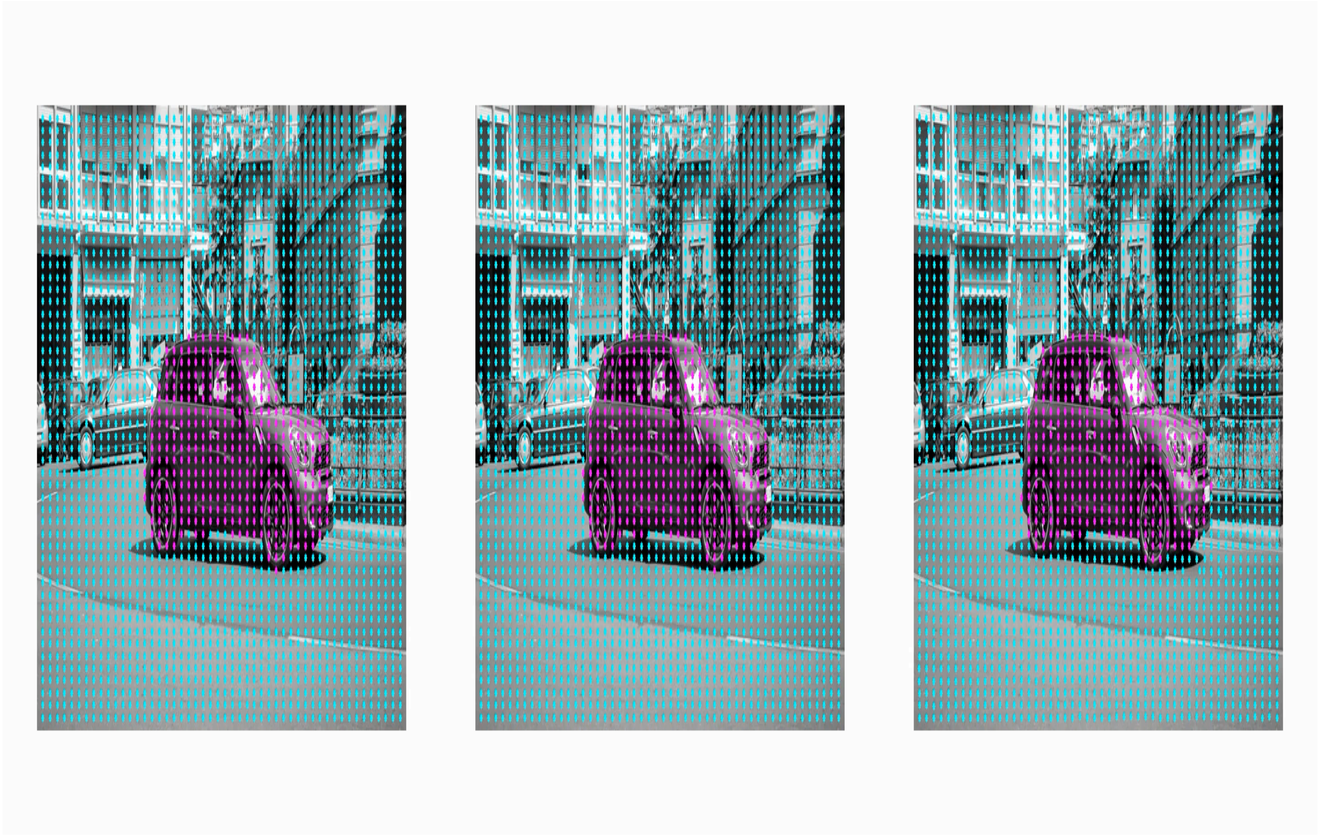
Tracking Points Together



- Background points
- Object points

- Better global consistency
- Better occlusion handling

Tracking Points Together



- Background points
- Object points

- Better global consistency
- Better occlusion handling

Tracking Points Together

Tracking Points Together

Tracking Points Together

Tracking Points Together

Flow Fields in 2D and 3D

Optical Flow

Optical Flow

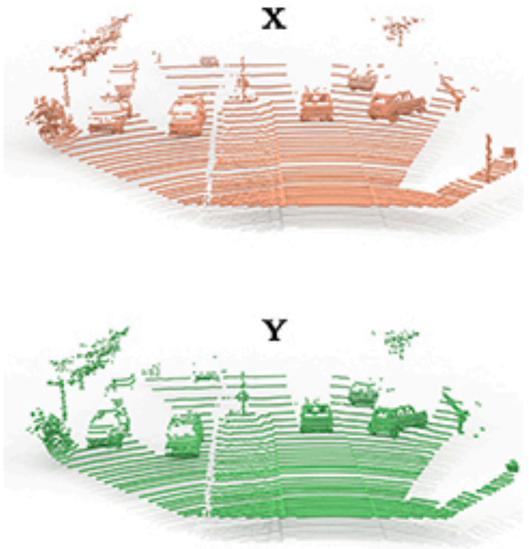
Given a pixel in frame 1, where does it go to in frame 2?

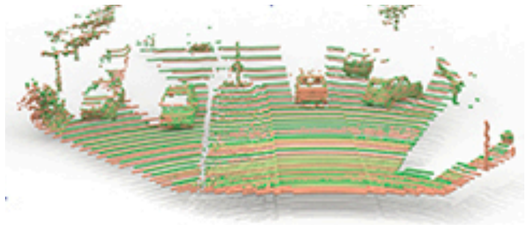
$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ A 2D flow field

Flow Fields in 2D and 3D

Optical Flow

Optical Flow





Per-point scene flow

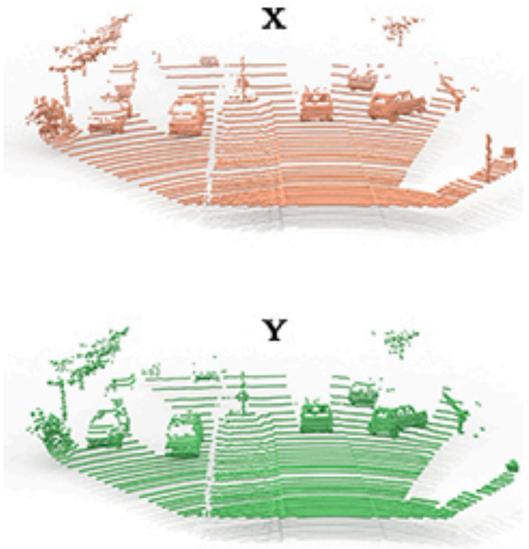
Given a pixel in frame 1, where does it go to in frame 2?

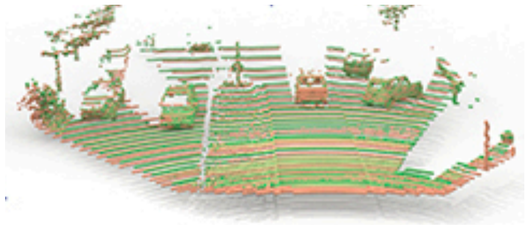
$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ A 2D flow field

Flow Fields in 2D and 3D

Optical Flow

Optical Flow





Per-point scene flow

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

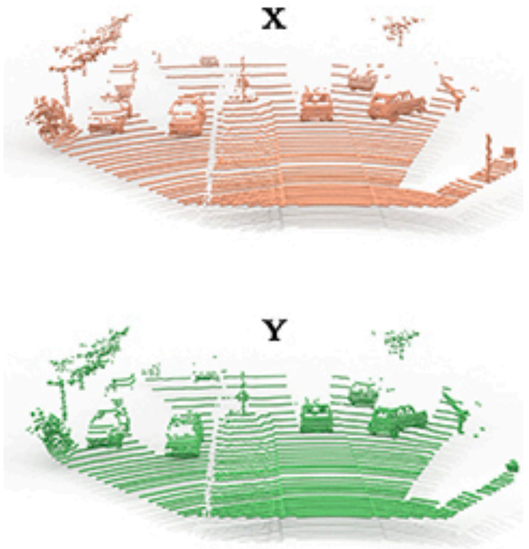
$\mathbb{R}^2 \rightarrow \mathbb{R}^2$ A 2D flow field

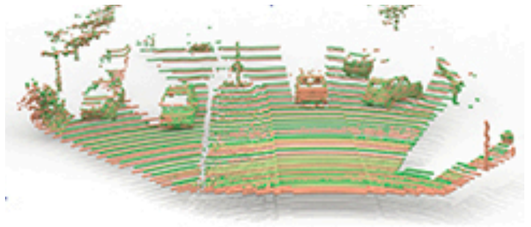
Given a 3D point at time t1, where does it go at time t2?

Flow Fields in 2D and 3D

Optical Flow

Optical Flow





Per-point scene 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

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

Flow Fields in 2D and 3D

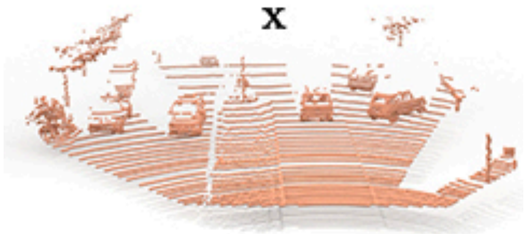
Optical Flow

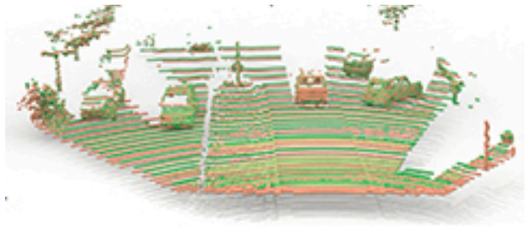
Optical Flow

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



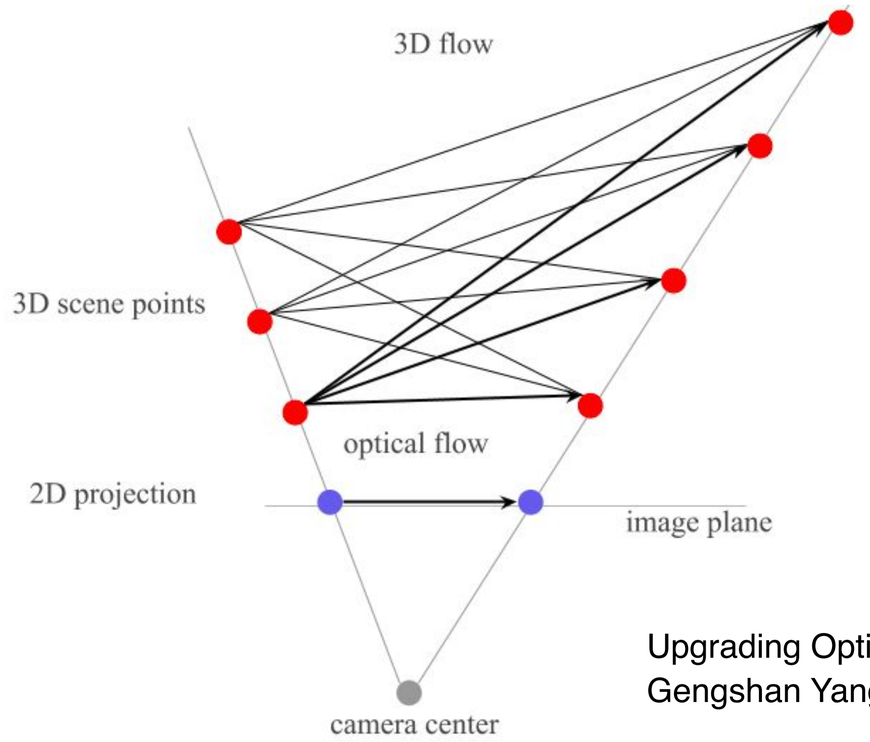


Per-point scene flow

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

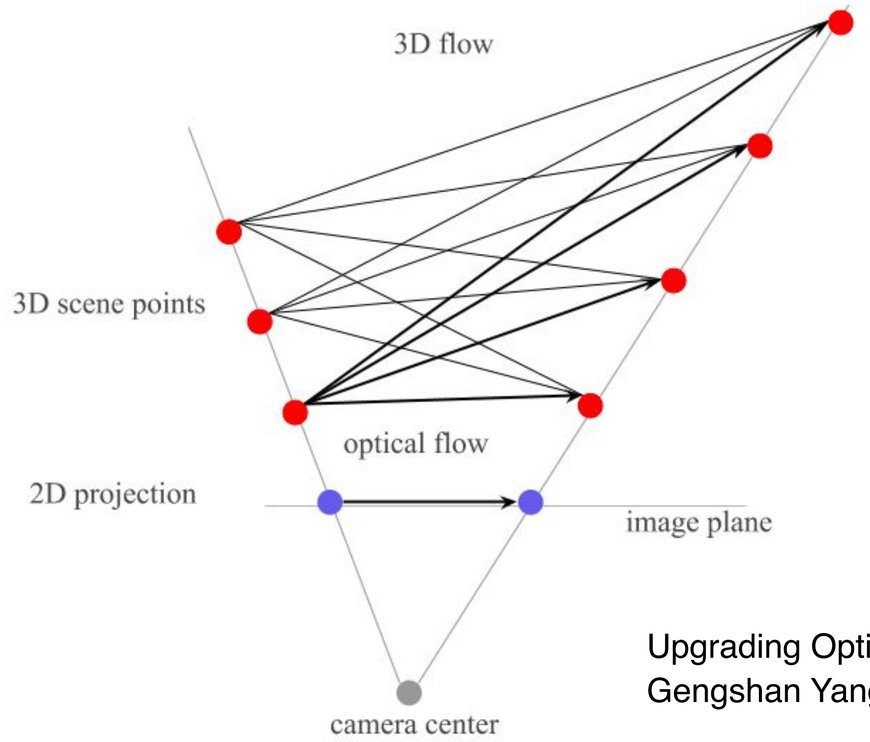
$\mathbb{R}^3 \rightarrow \mathbb{R}^3$ A 3D flow field

3D Scene Flow



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

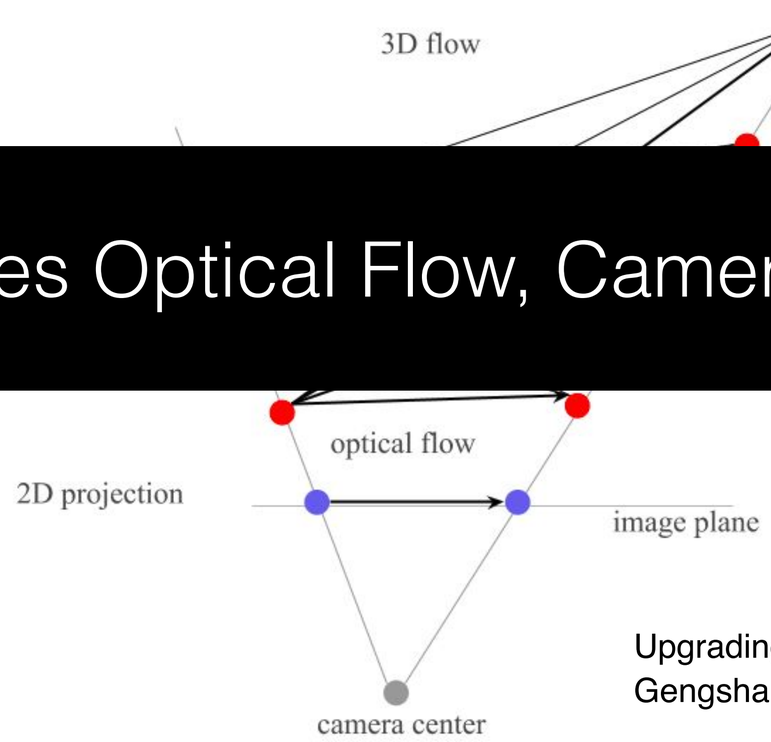
3D Scene Flow



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

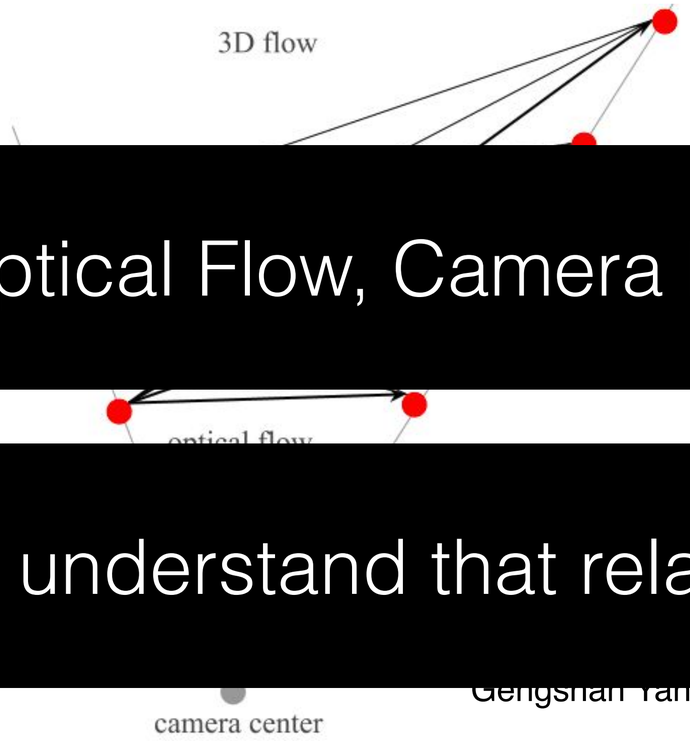


Scene Flow Relates Optical Flow, Camera Motion, and Depth!

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!

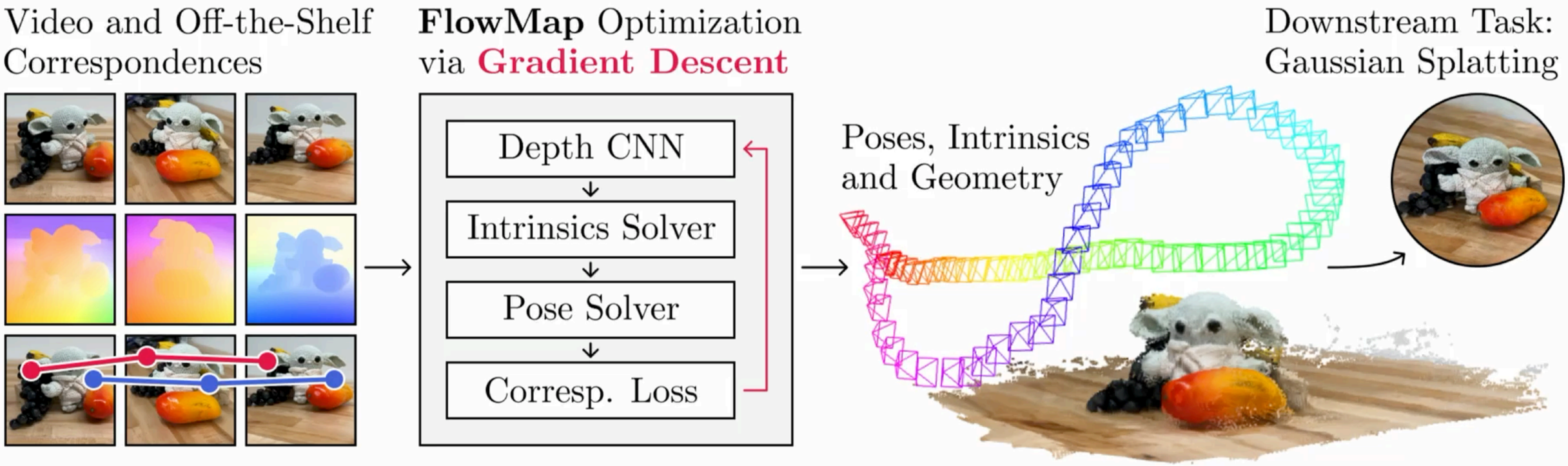
Let's deeply understand that relationship.

on,

Gengshan Yang et al, CVPR 2020

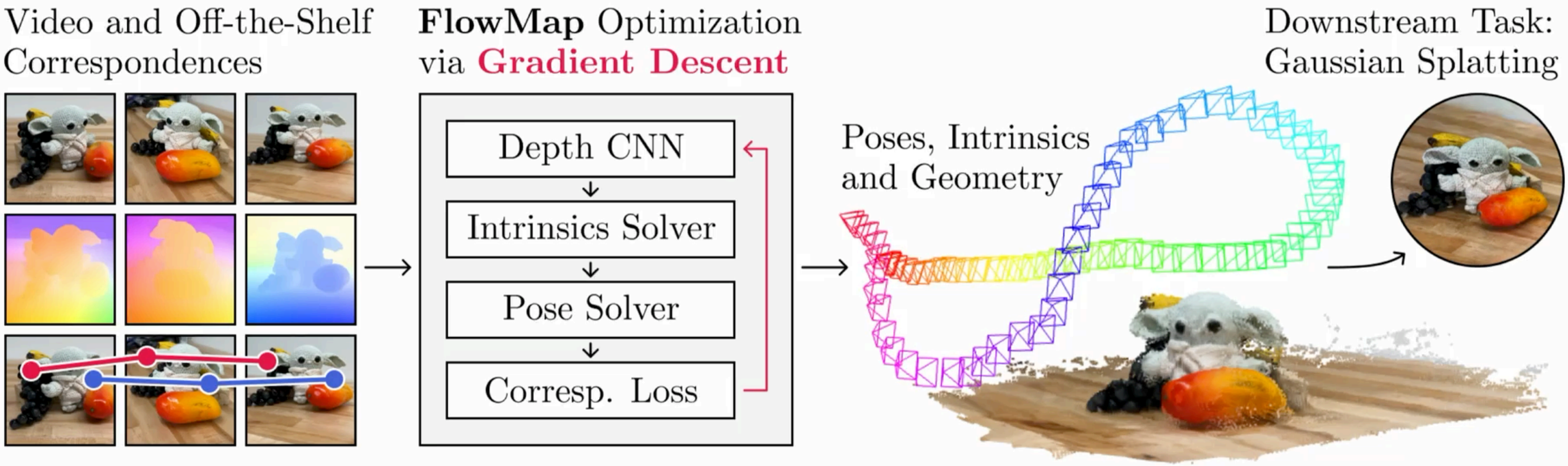
Observation: 2D flow constrains possible 3D flow (for visible points), and optical flow can be computed given 3D scene flow plus depth

FlowMap: Differentiable Structure-from-Motion via Gradient Descent



Gaussian Splats on par with COLMAP's!

FlowMap: Differentiable Structure-from-Motion via Gradient Descent

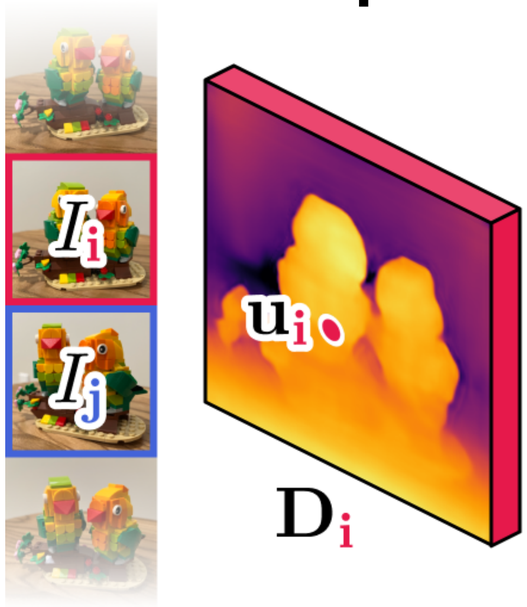


Gaussian Splats on par with COLMAP's!

The Camera-Induced Flow Loss

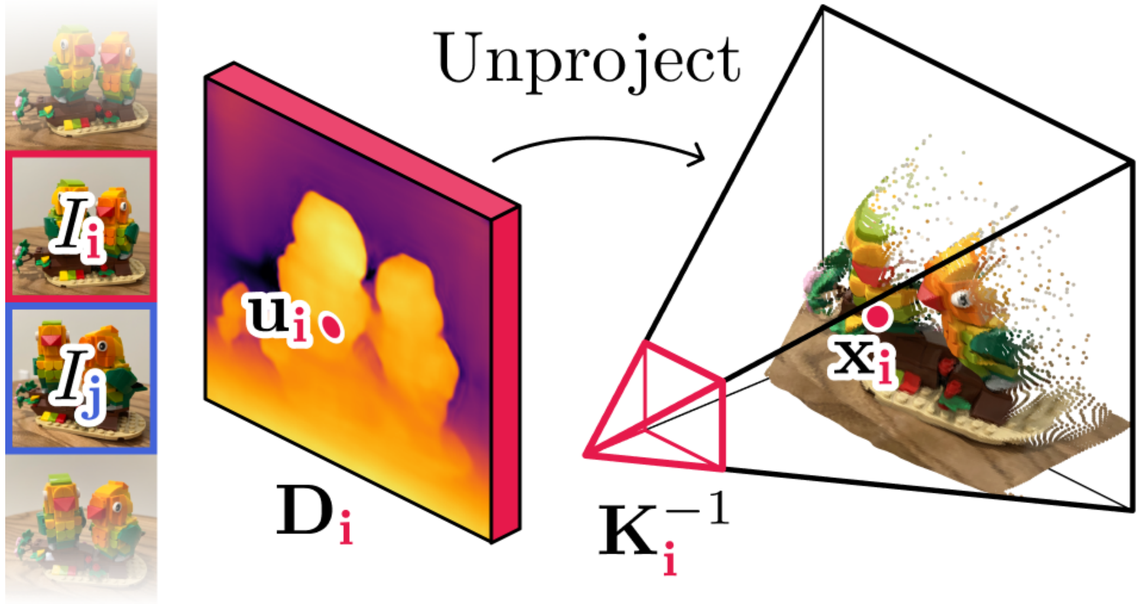
I_i I_j

The Camera-Induced Flow Loss



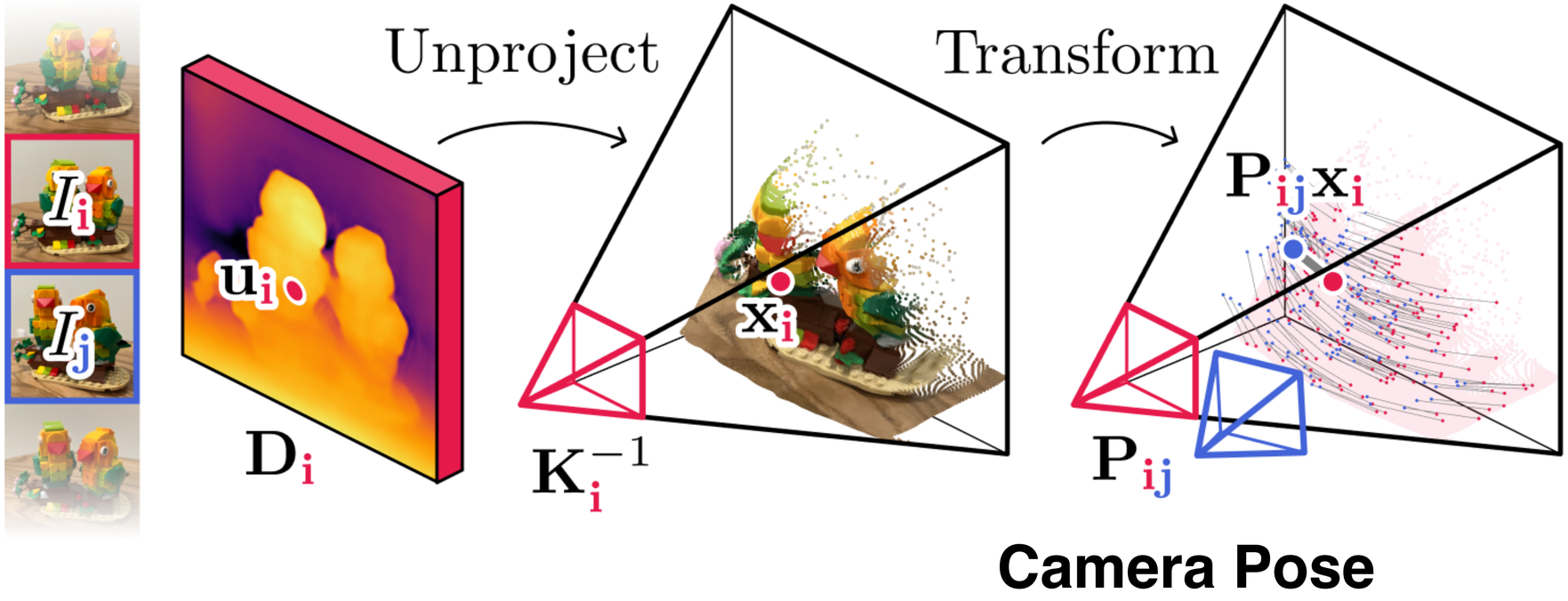
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss



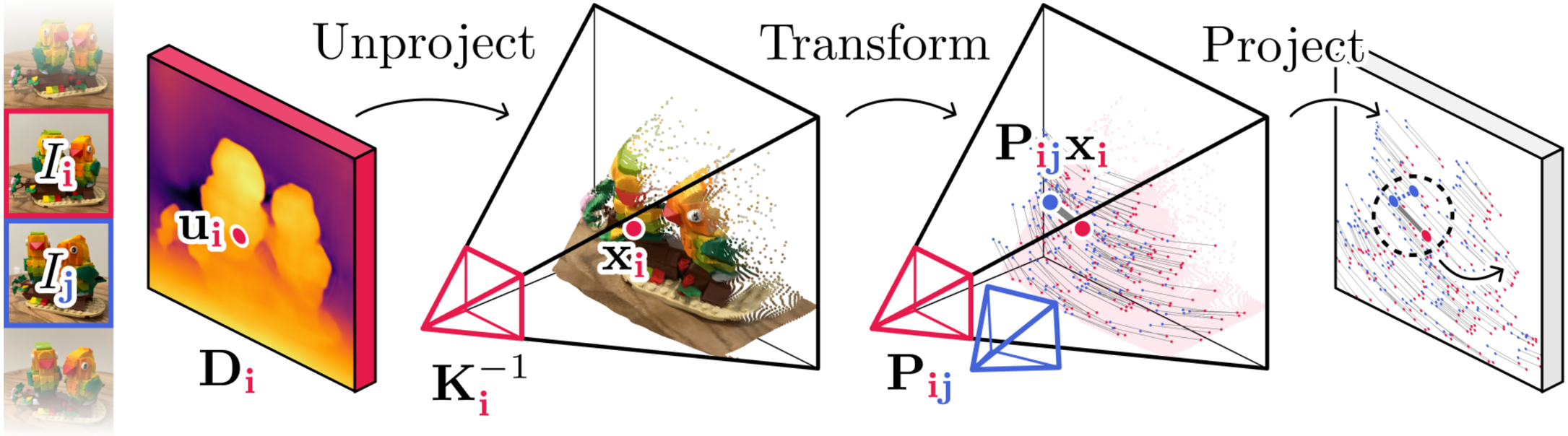
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss



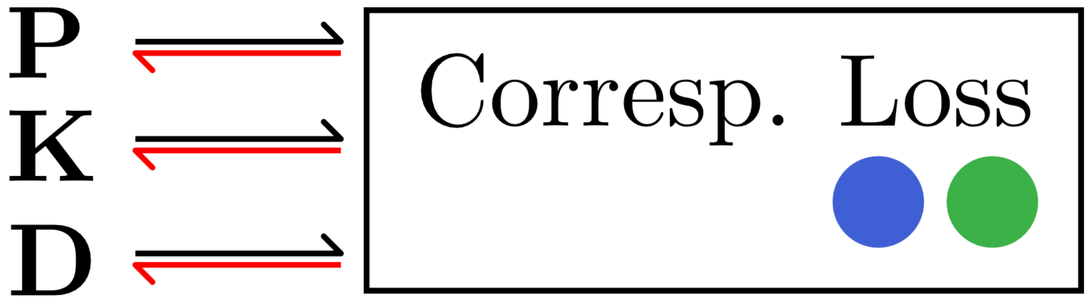
Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

The Camera-Induced Flow Loss

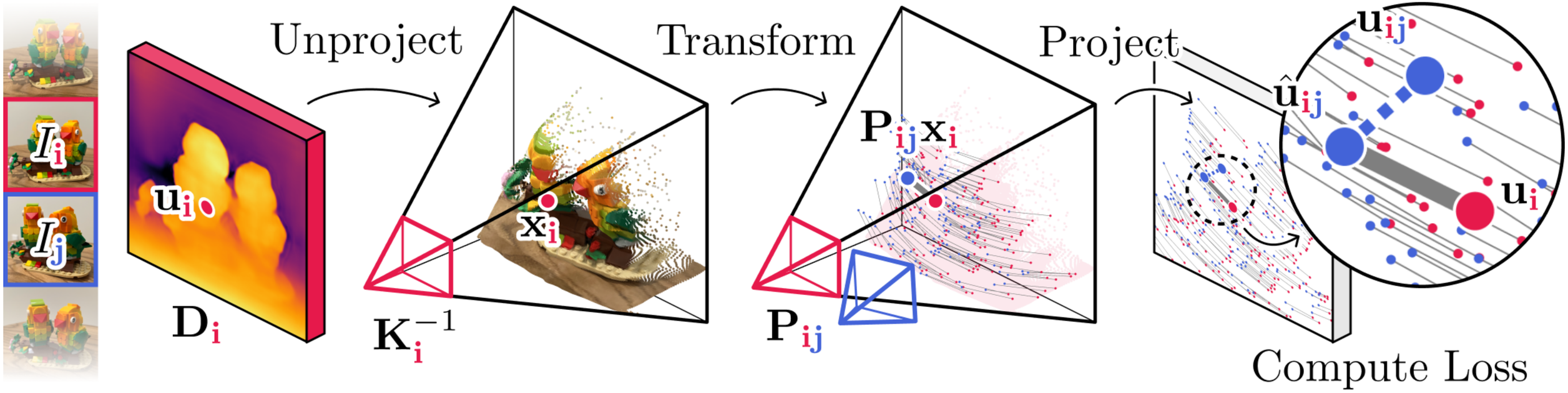


Smith et al: FlowMap: High-Quality Camera Poses, Intrinsics, and Depth via Gradient Descent; 3DV 2025

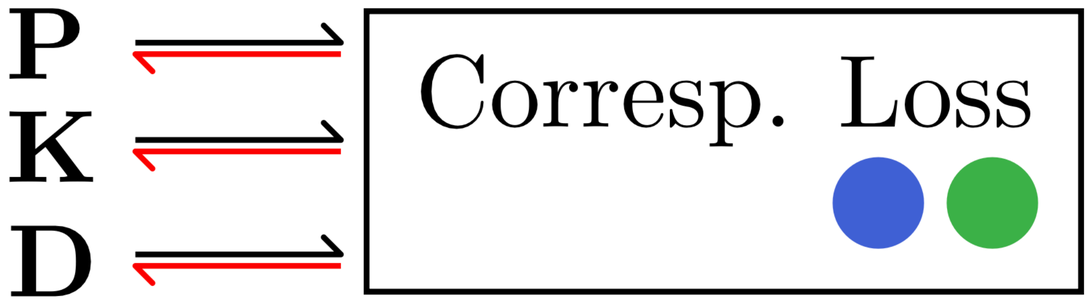
The Camera-Induced Flow Loss



The Camera-Induced Flow Loss



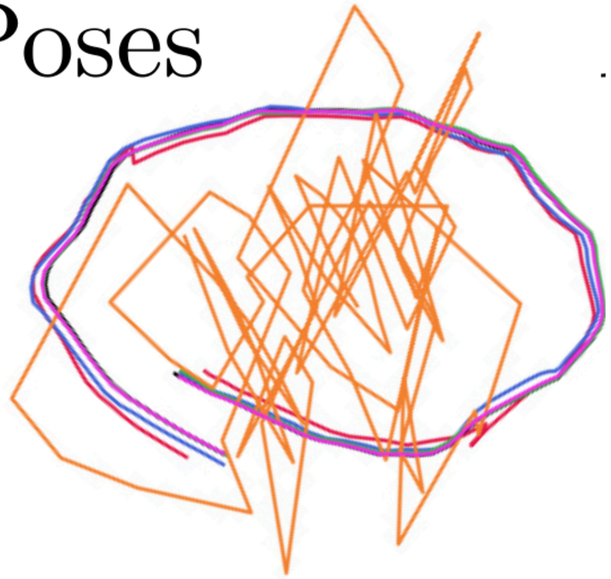
Let’s optimize depth, pose and intrinsics directly!



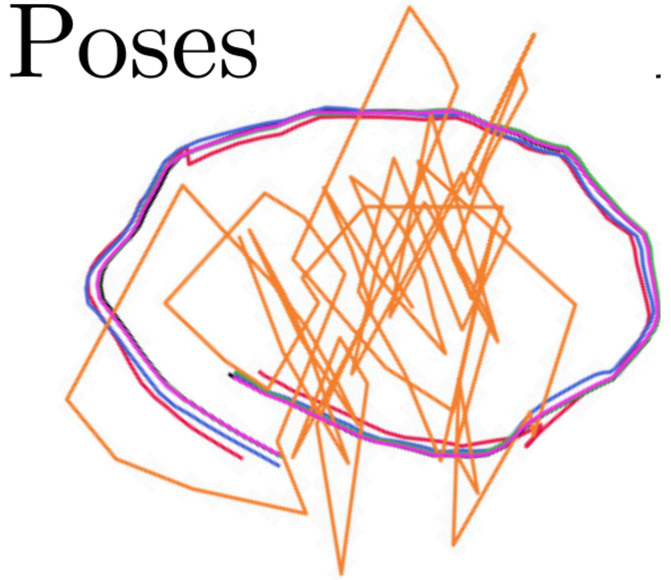
Doesn’t work :/

Pose Depth Focal

Poses

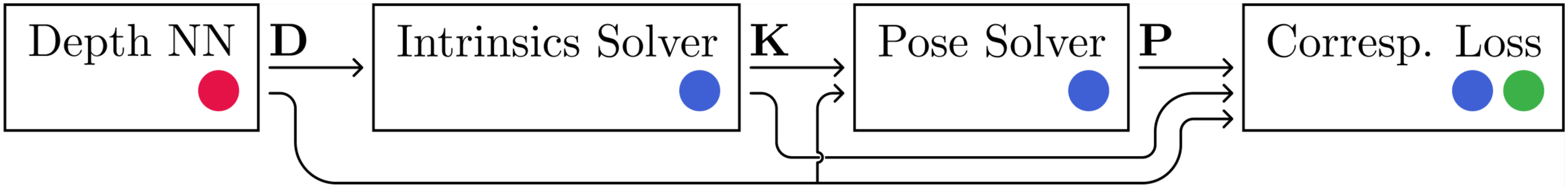


Doesn’t work :/

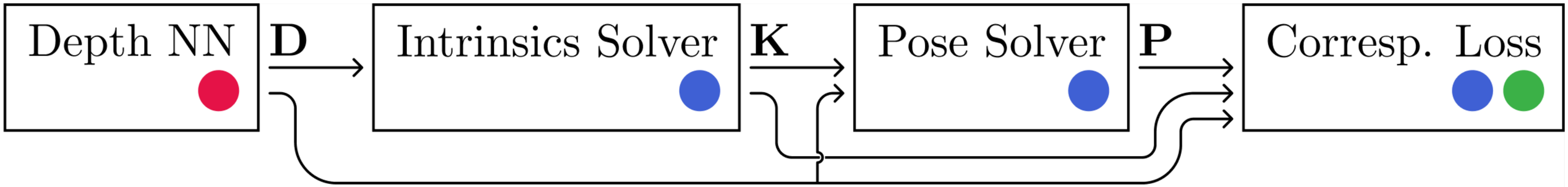


Why not? Lots of local minima: Hard to “get unstuck” from a wrong pose or wrong intrinsics, plus lots of ways for model to “cheat”

FlowMap: Feed-Forward Estimators!

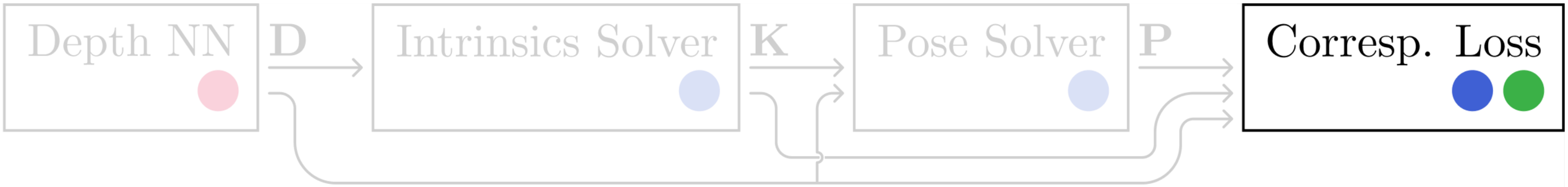


FlowMap: Differentiable SfM via Gradient Descent



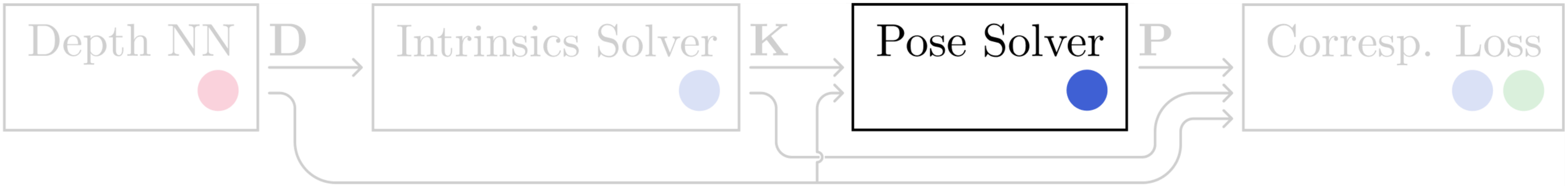
RGB Flow Tracks

FlowMap: Differentiable SfM via Gradient Descent

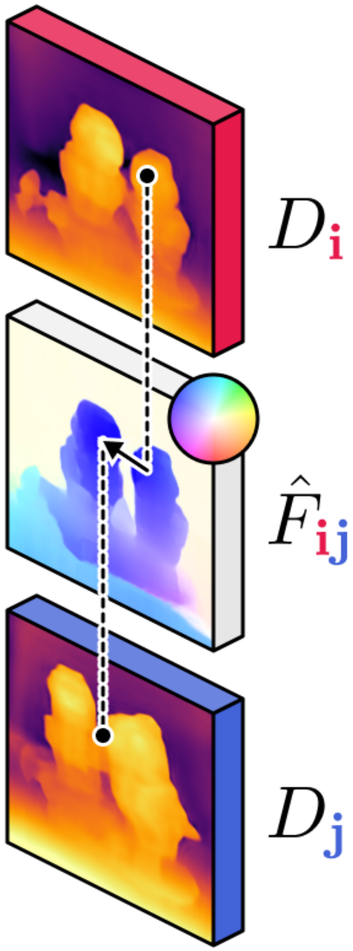


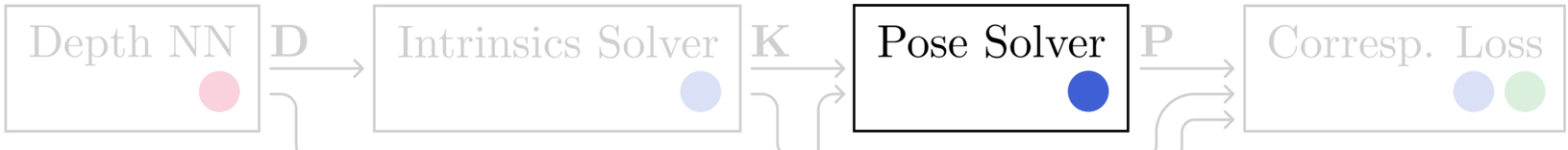
RGB Flow Tracks

FlowMap: Differentiable SfM via Gradient Descent

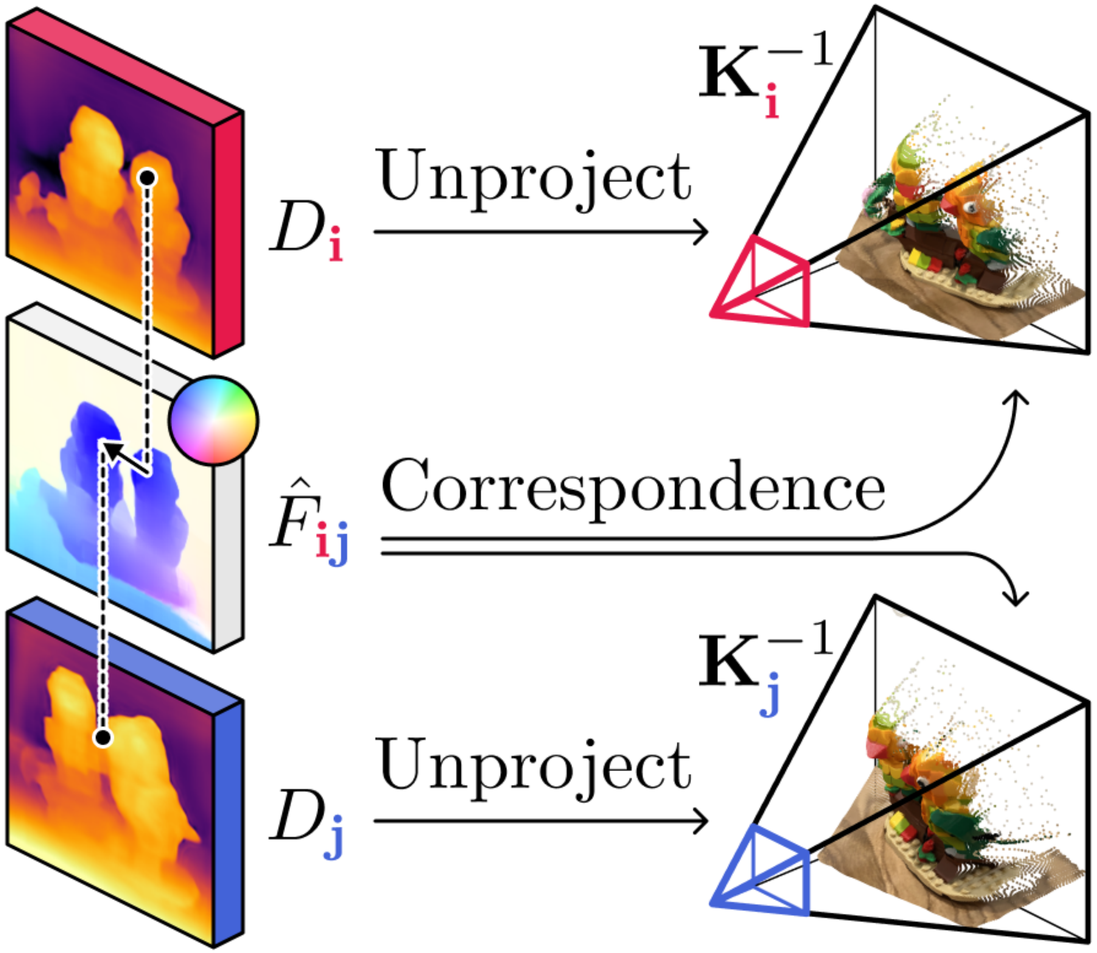


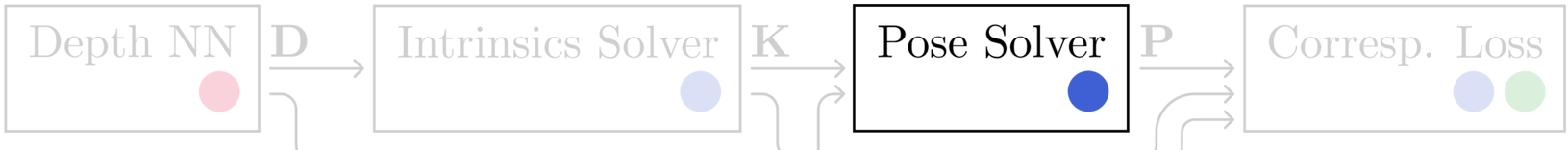
Diff. Least-Squares Pose Solve



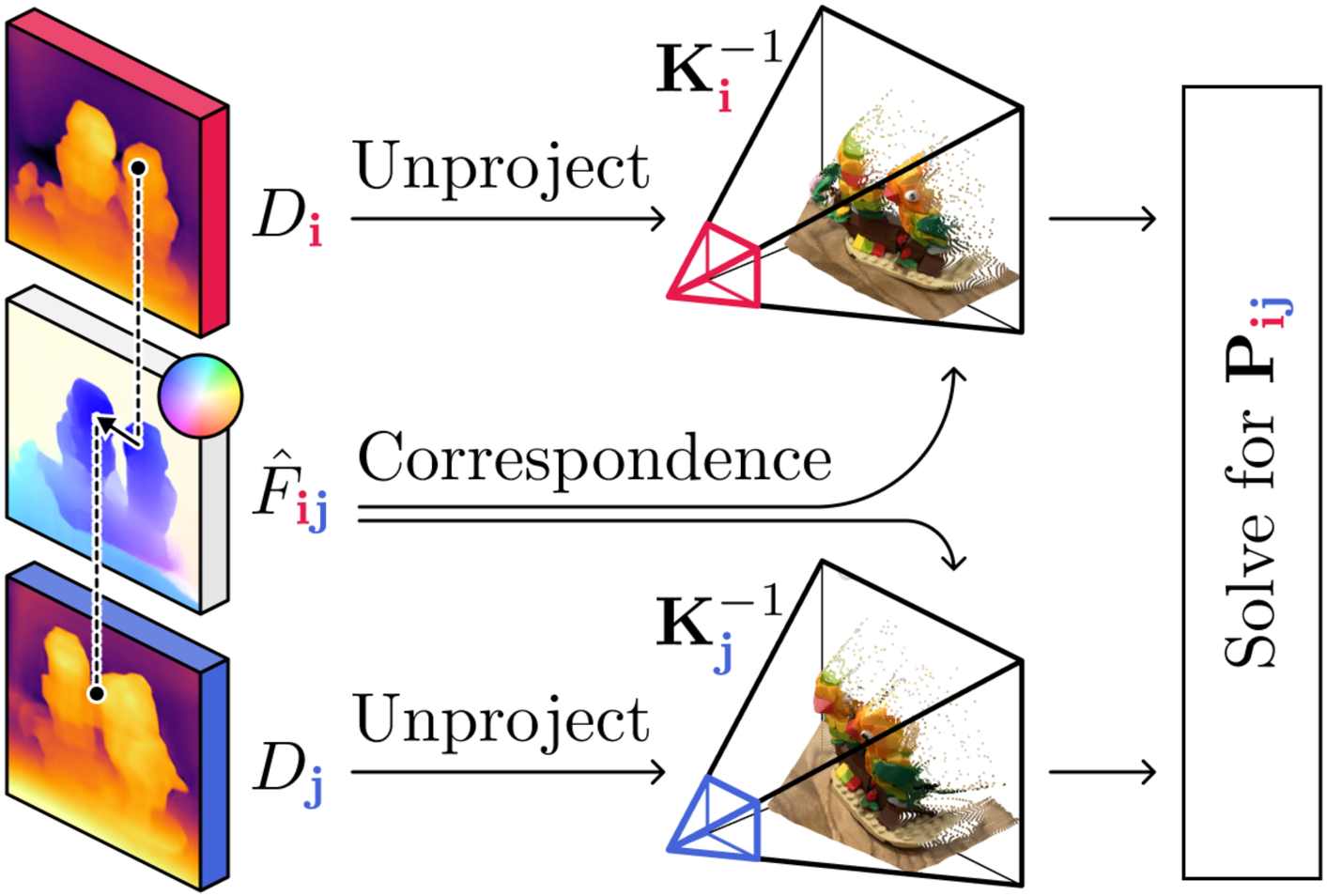


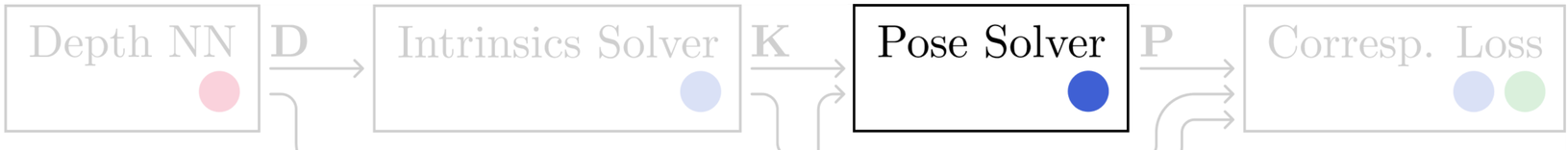
Diff. Least-Squares Pose Solve



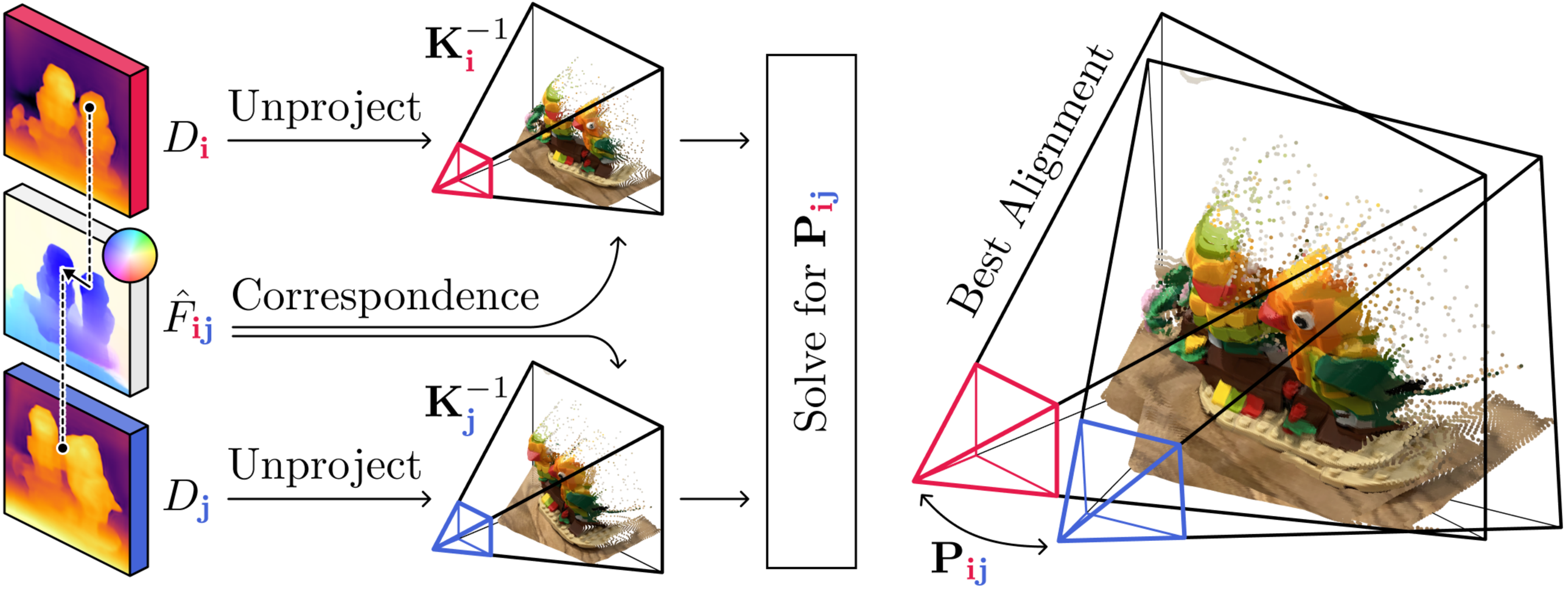


Diff. Least-Squares Pose Solve

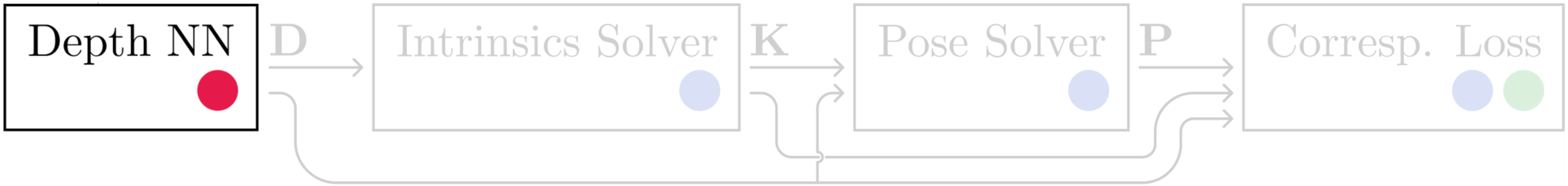




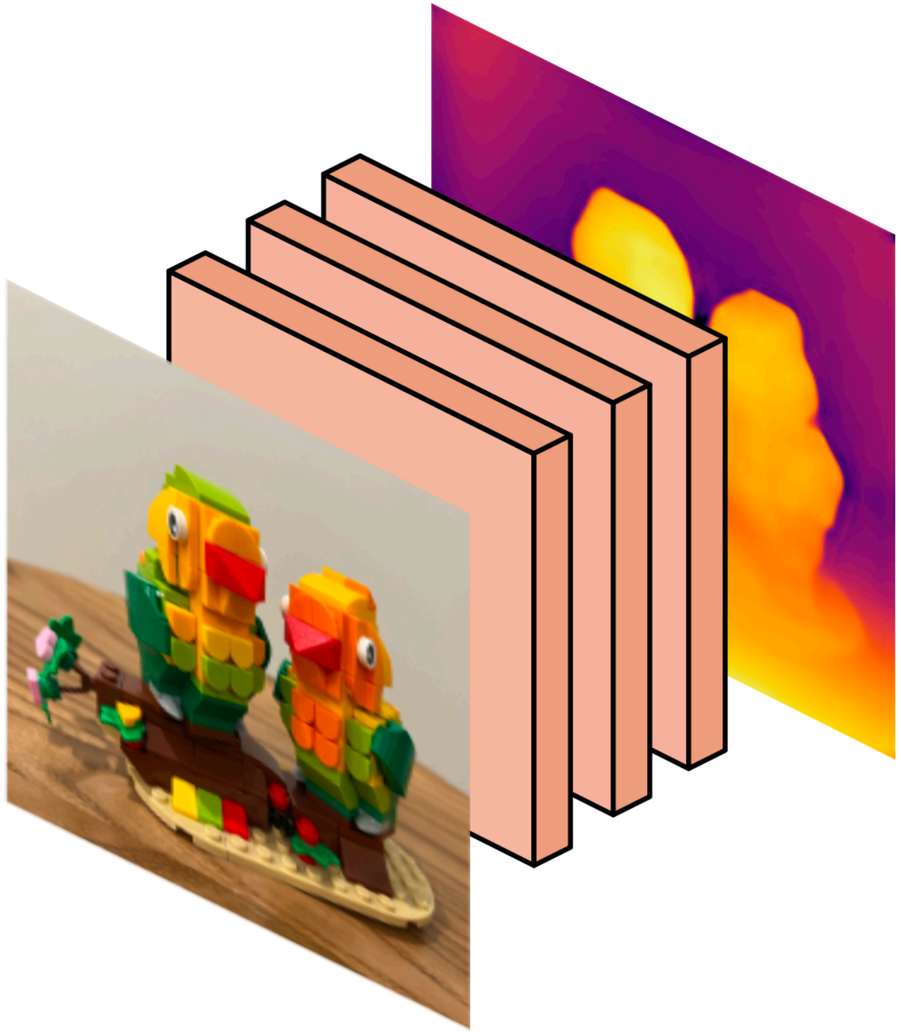
Diff. Least-Squares Pose Solve



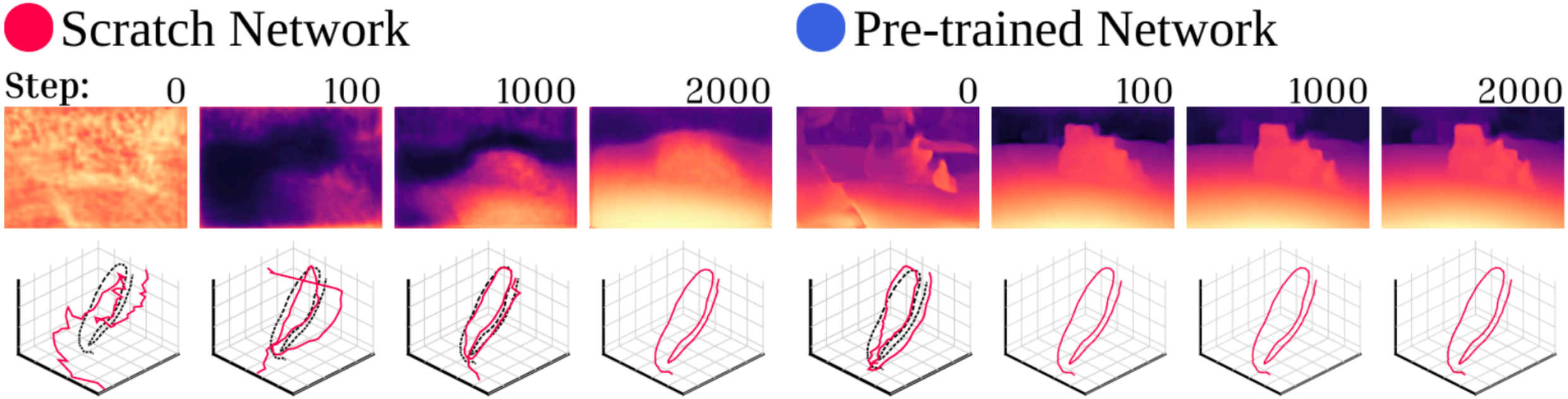
Predict Depth from RGB



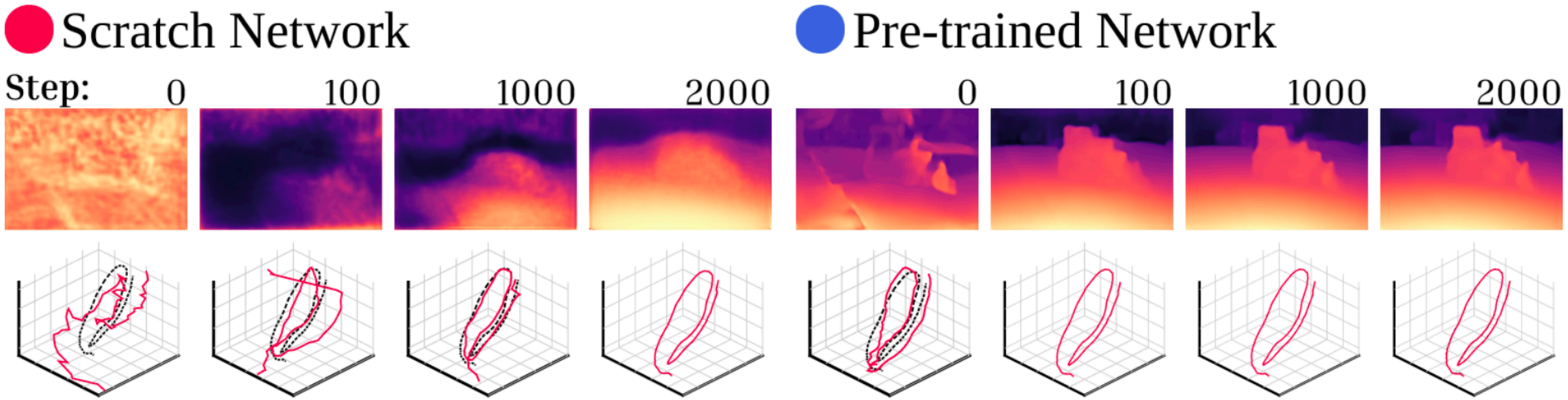
Depth Network



Depth Network Need Not be Pre-trained!

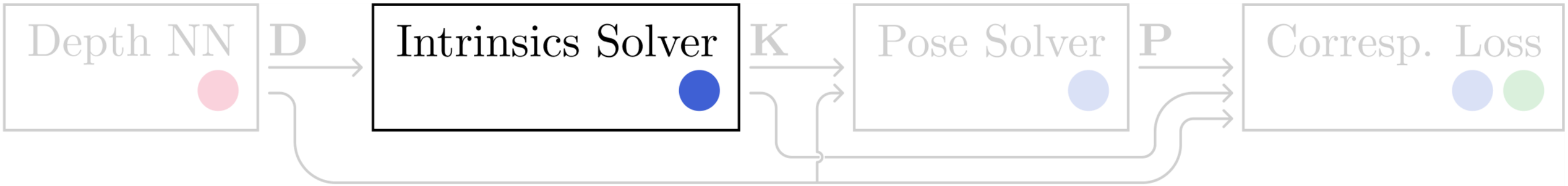


Depth Network Need Not be Pre-trained!

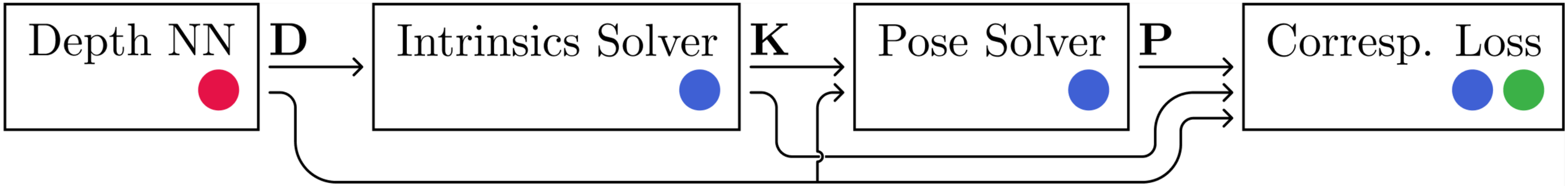


- Depth CNN acts as a **patch-match** regularizer: same RGB patch leads to same depth!
- But: **Can be pre-trained end-to-end on large video datasets without annotations, leading to much faster convergence!**
- In the limit of perfect monocular depth estimator: Feed-Forward SfM!

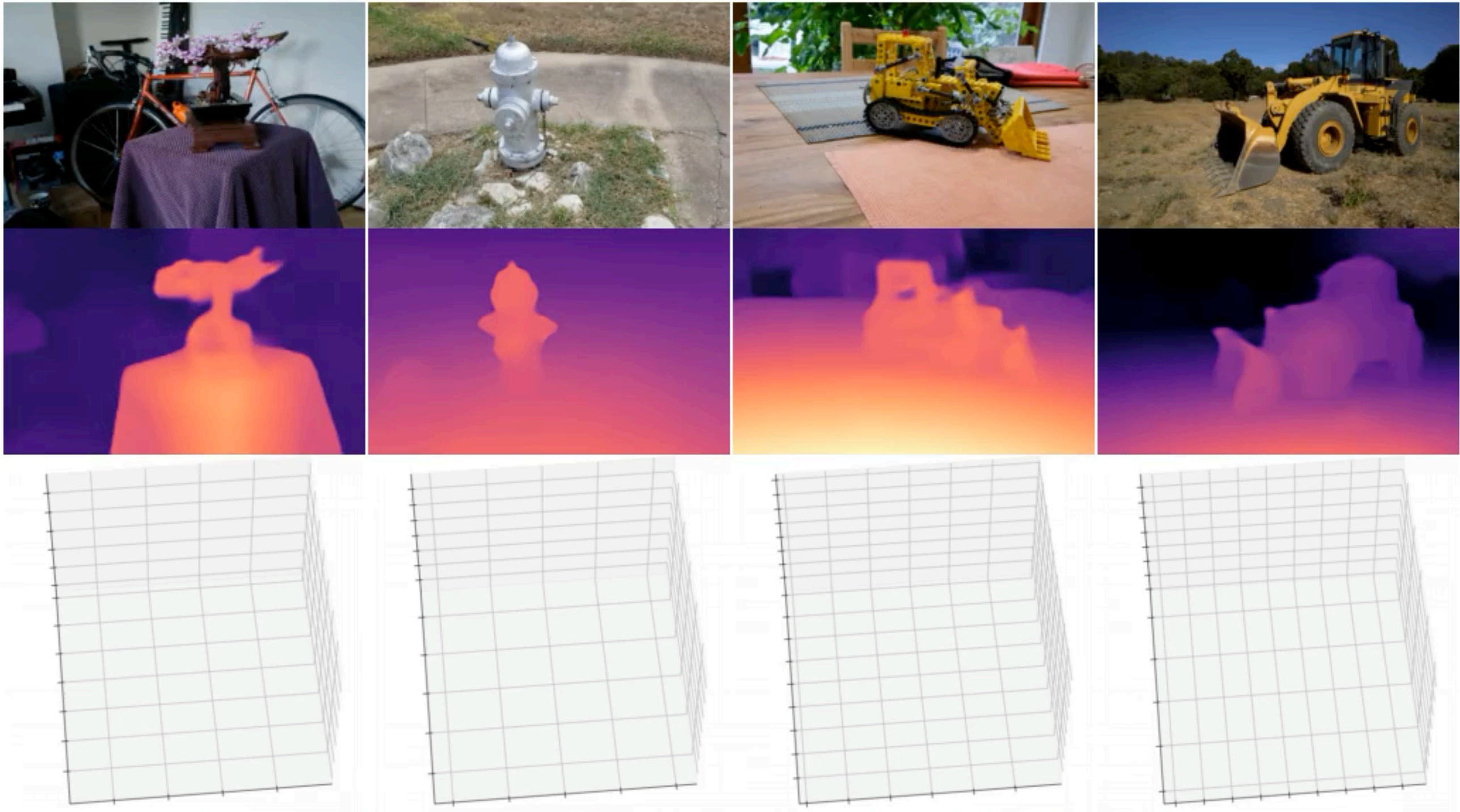
No time to talk about diff. Intrinsics solver - pls look at paper!

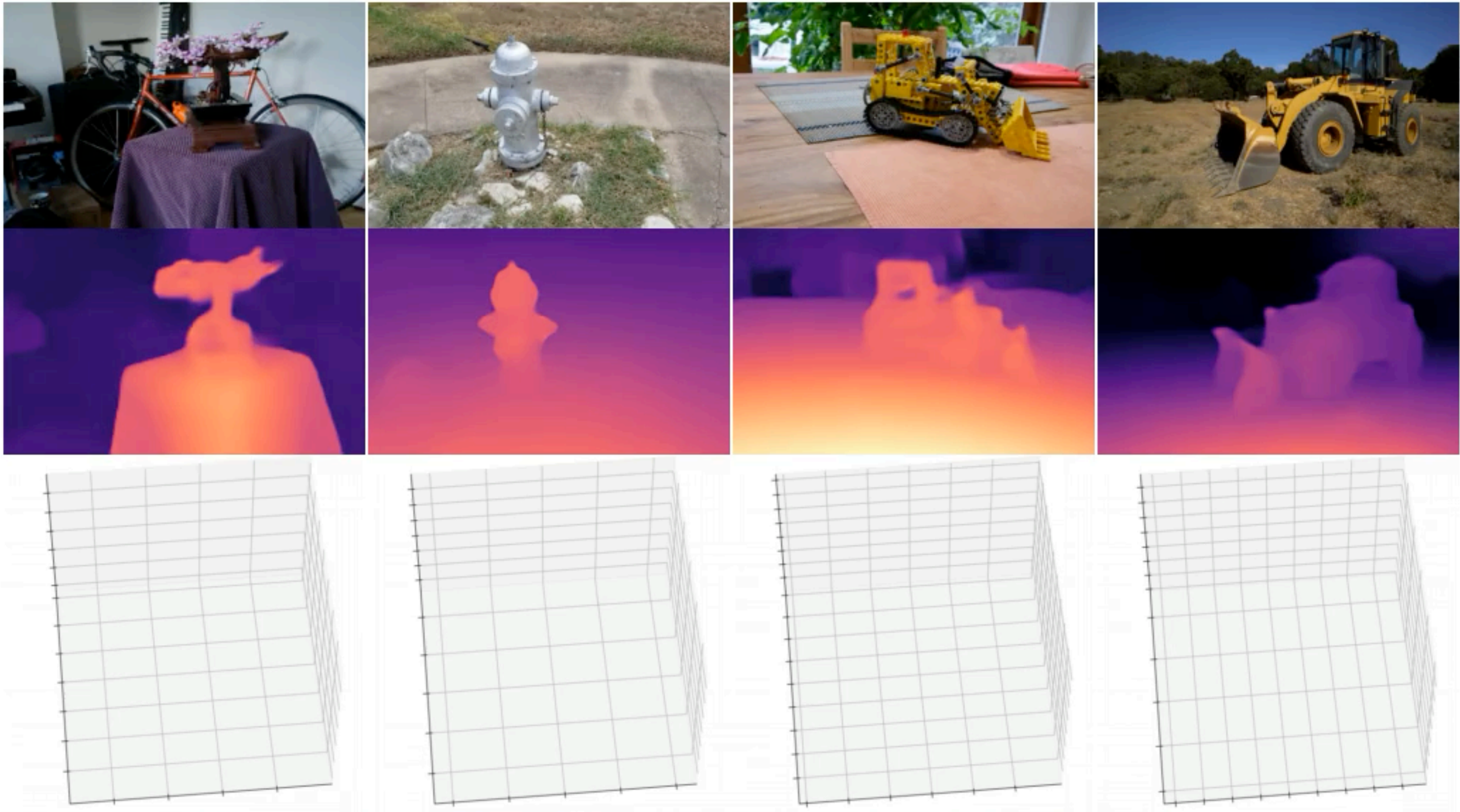


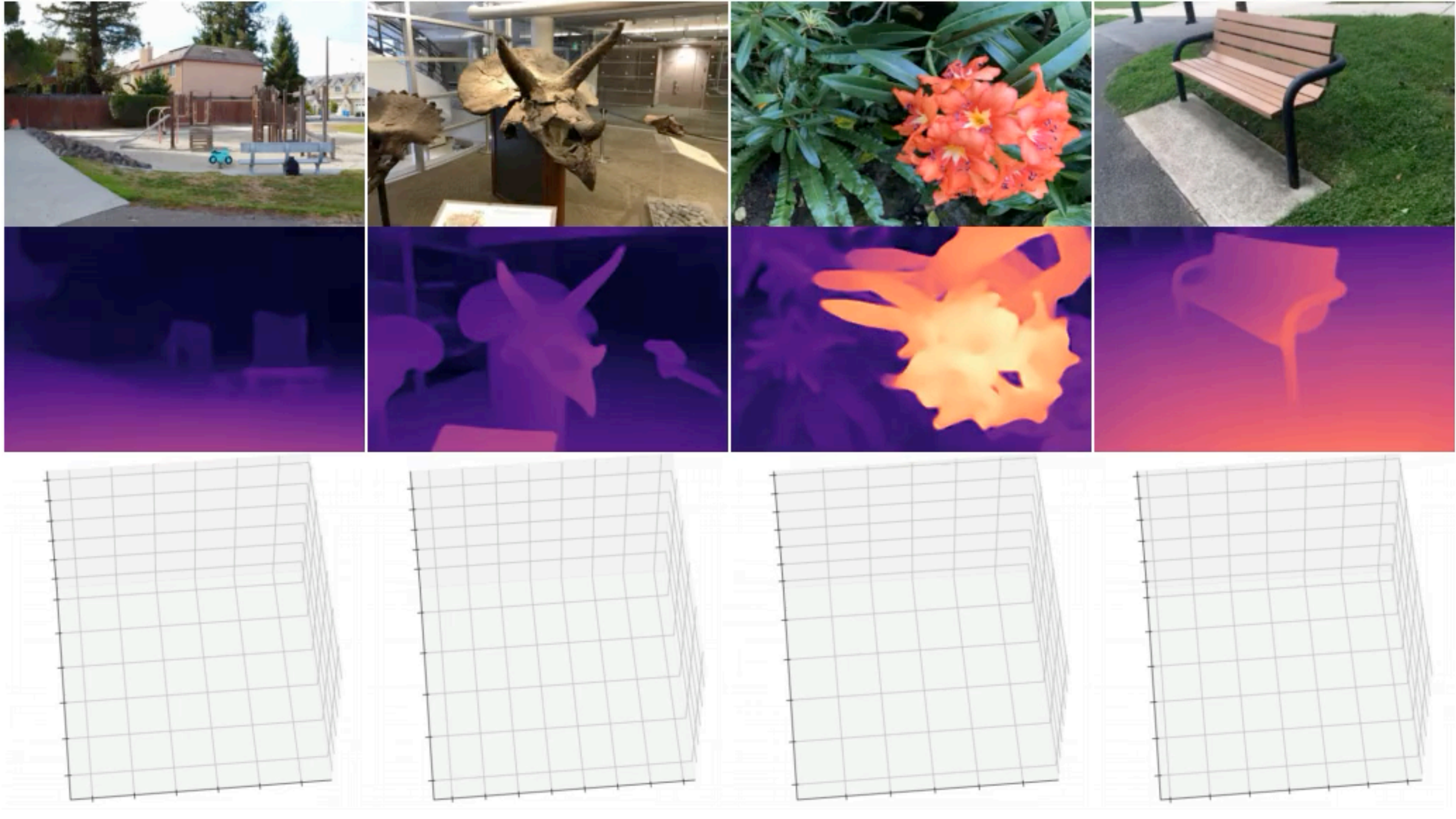
FlowMap: Differentiable SfM via Gradient Descent

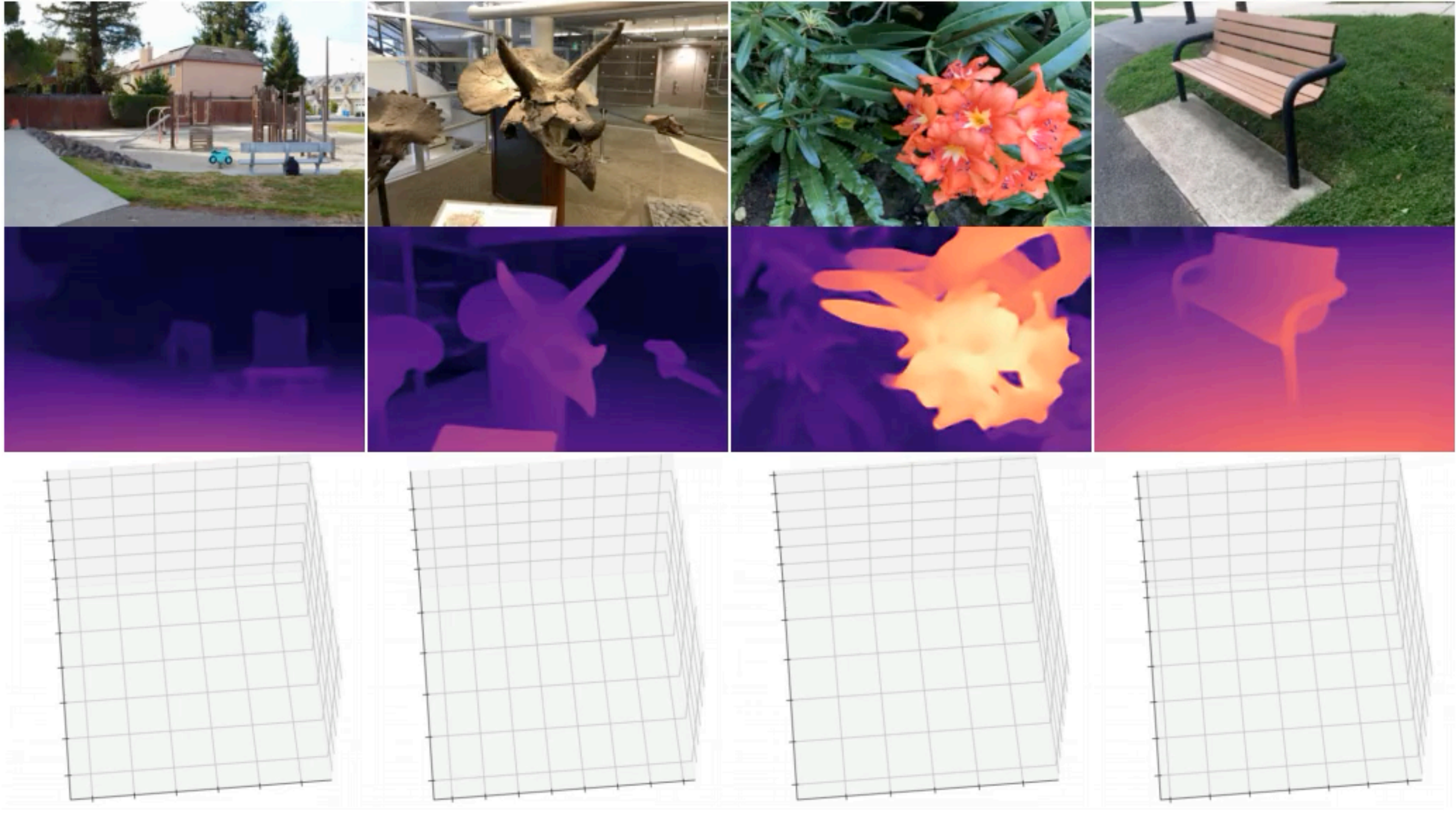


- FlowMap is fit end-to-end via gradient descent on each video separately.
- **Only free parameters are depth network and correspondence weights in pose solver!**
- These weights *can* be pre-trained, but *need not* be pre-trained: The model is constrained almost fully!

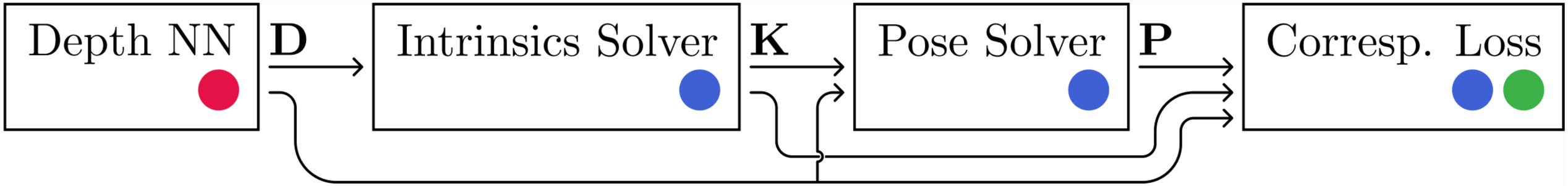








FlowMap: Differentiable SfM via Gradient Descent



- 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!

3D Scene Flow

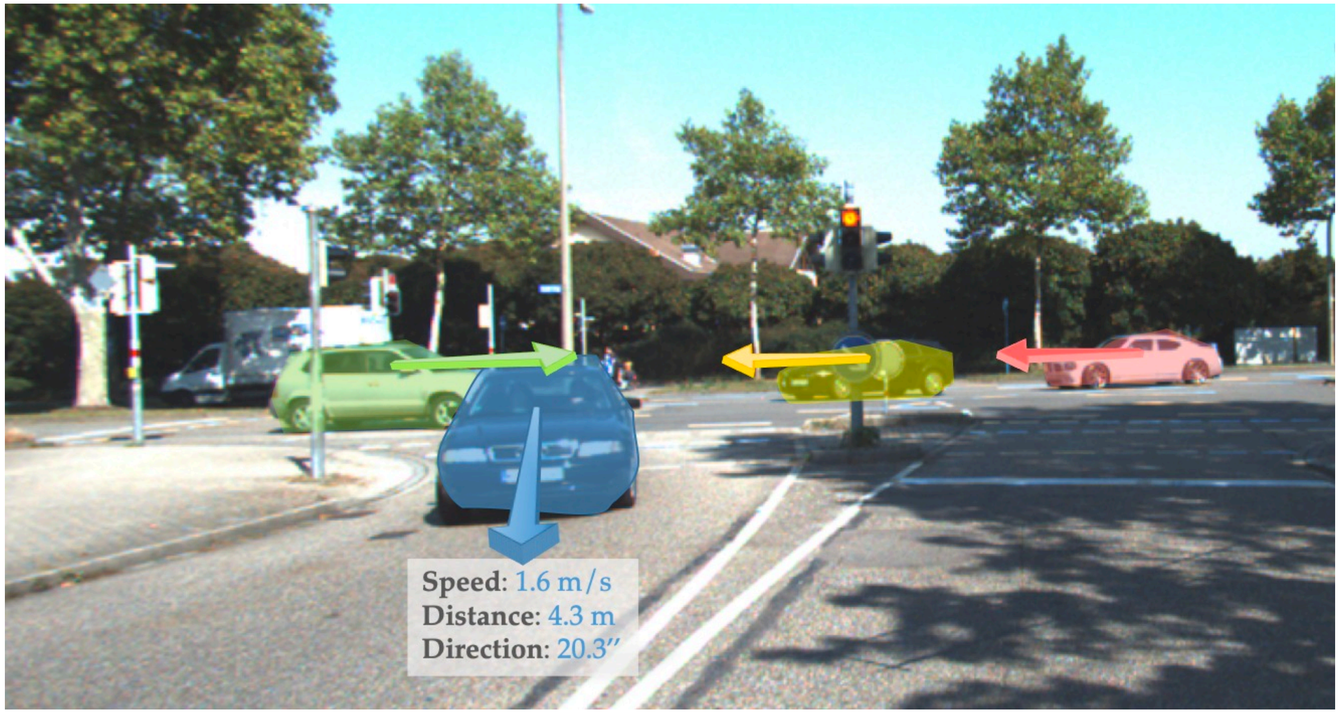
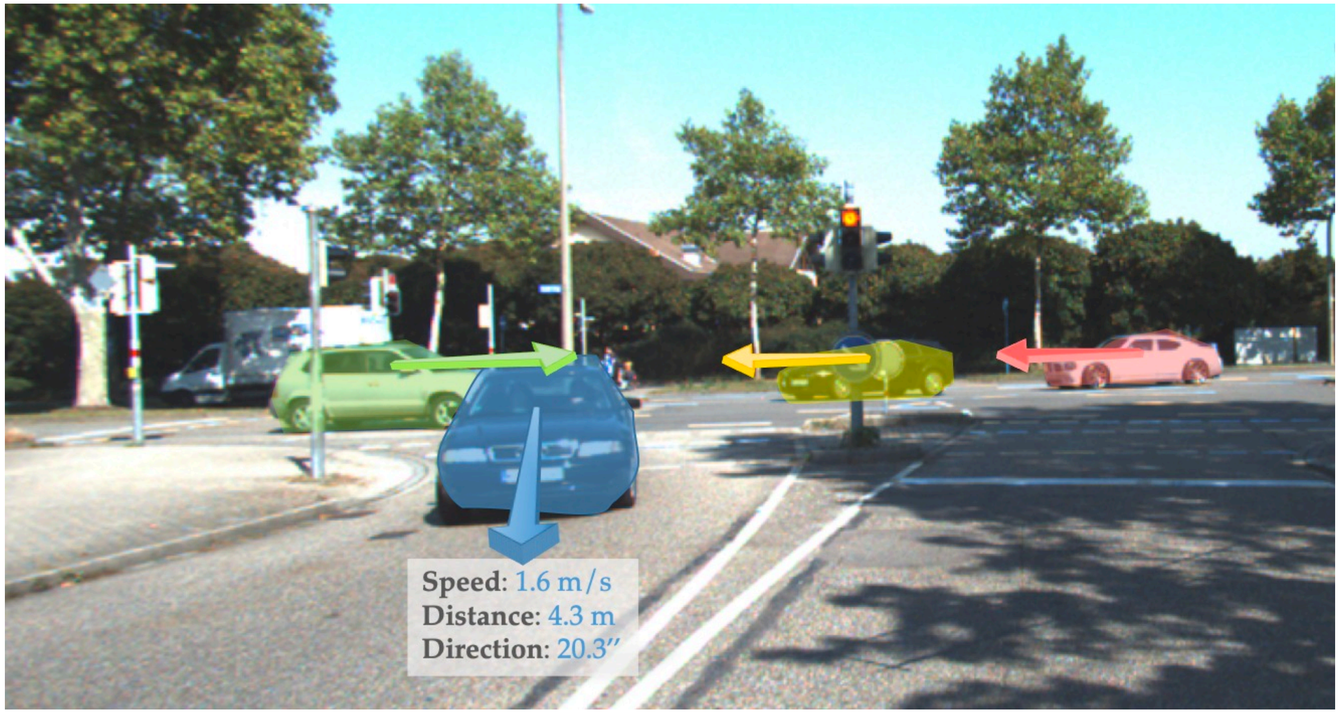


Image Courtesy: Zhe Cao

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

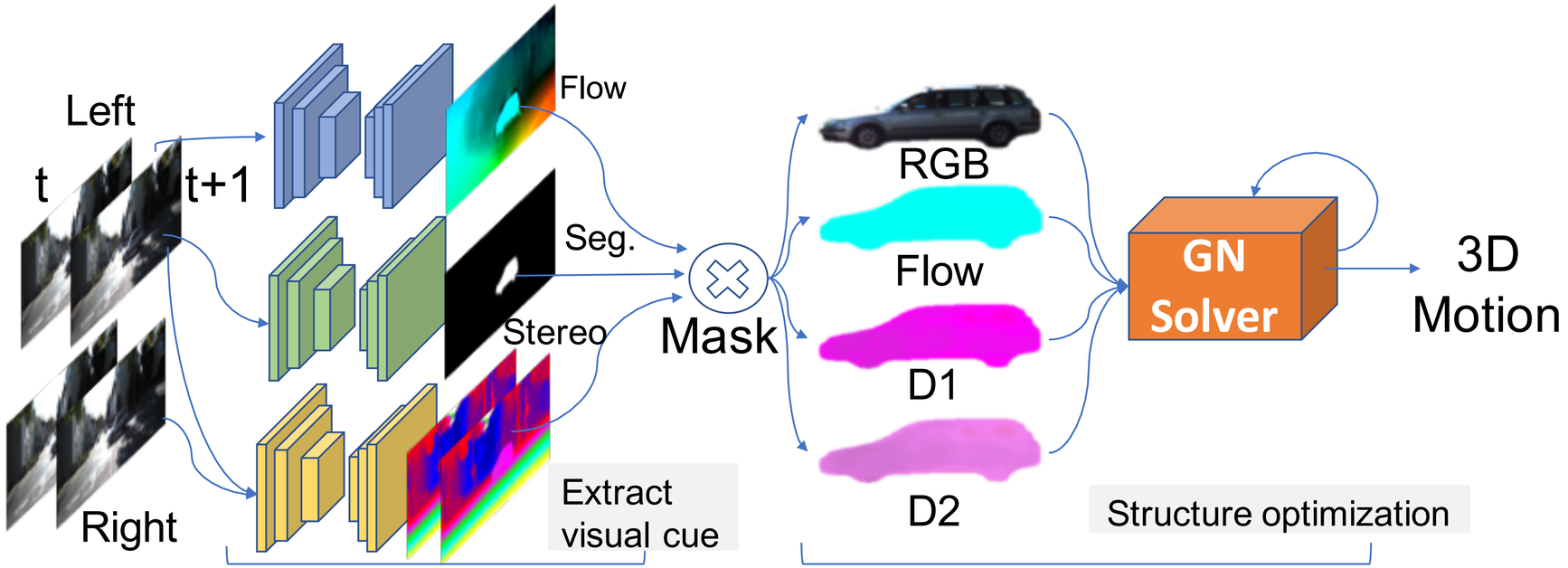


Figure 2: **Overview of our approach:** Given two consecutive stereo images, we first estimate the flow, stereo, and segmentation (Sec. 3.1). The visual cues of each instance are then encoded as energy functions (Sec. 3.2) and passed into the Gaussian-Newton (GN) solver to find the best 3D rigid motion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

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

Predicting 3D Scene Flow

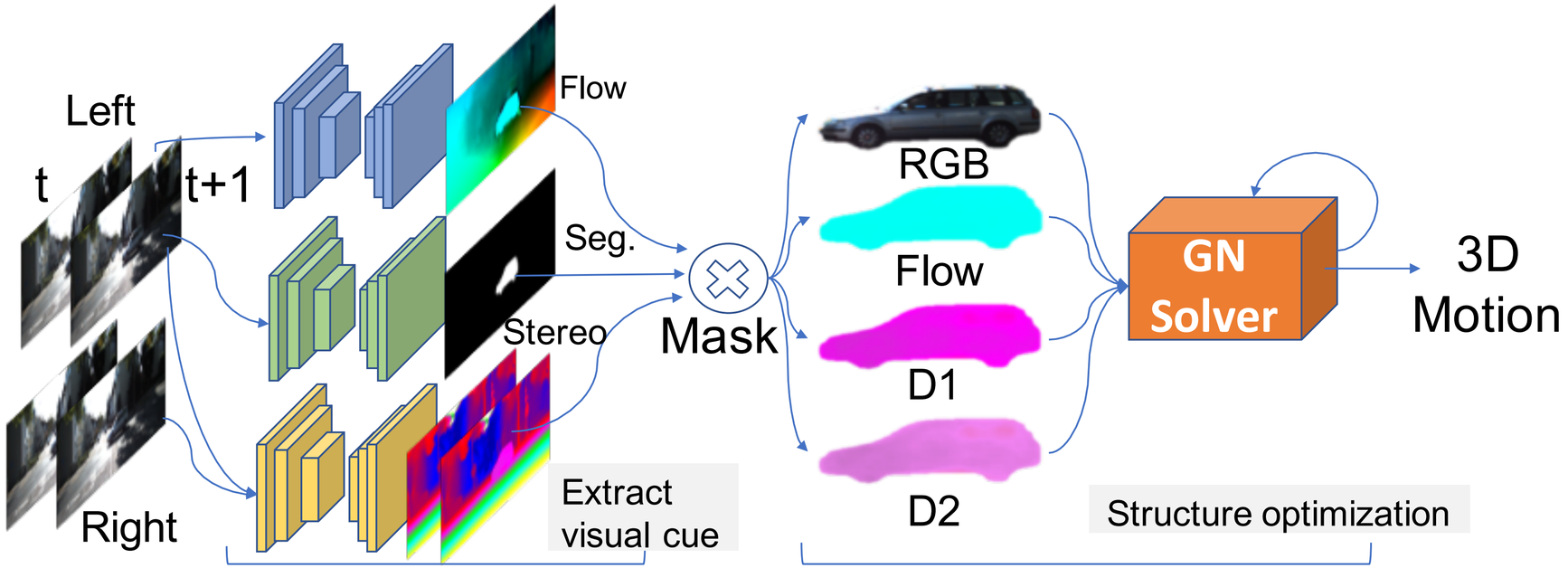


Figure 2: **Overview of our approach:** Given two consecutive stereo images, we first estimate the flow, stereo, and segmentation (Sec. 3.1). The visual cues of each instance are then encoded as energy functions (Sec. 3.2) and passed into the Gaussian-Newton (GN) solver to find the best 3D rigid motion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

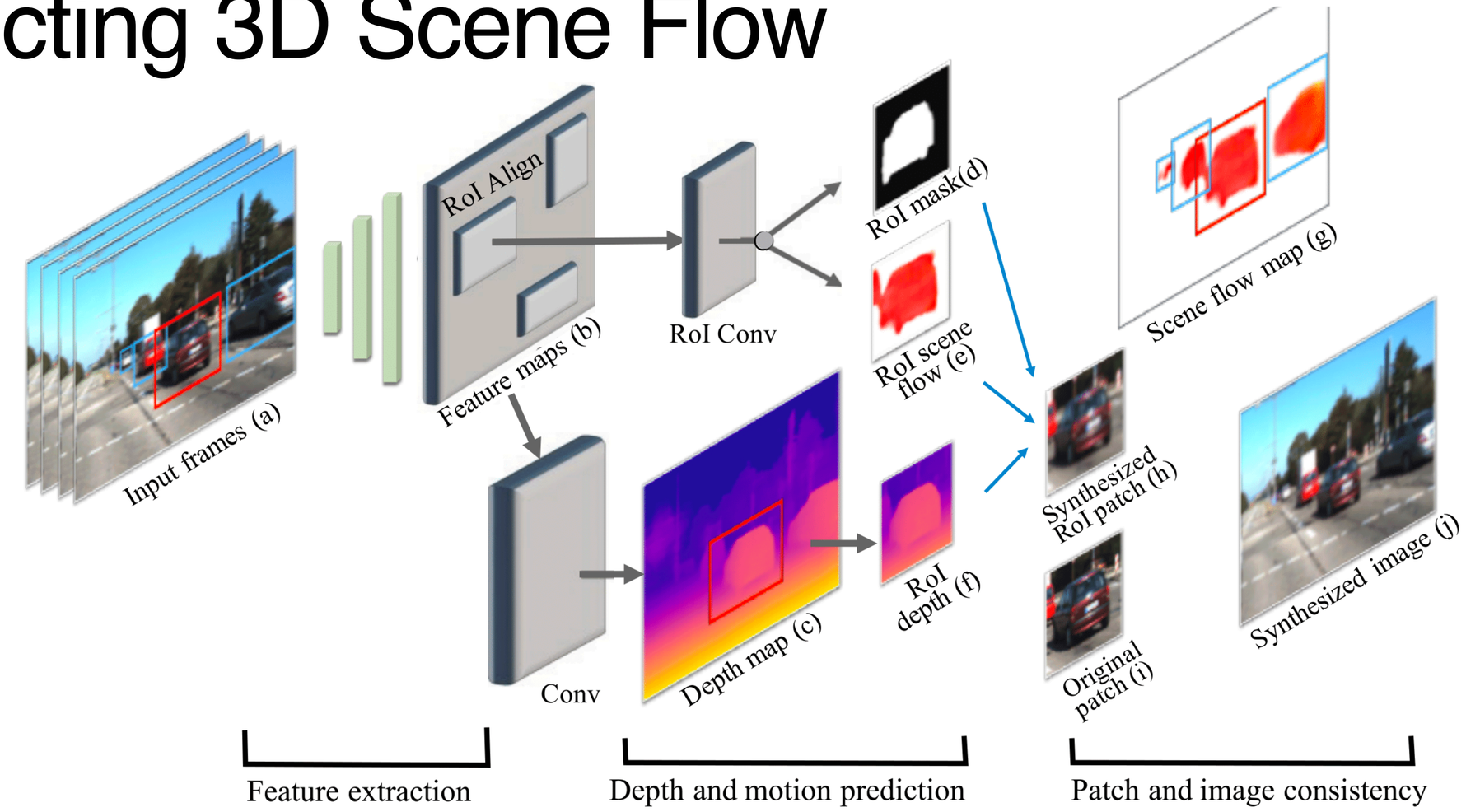
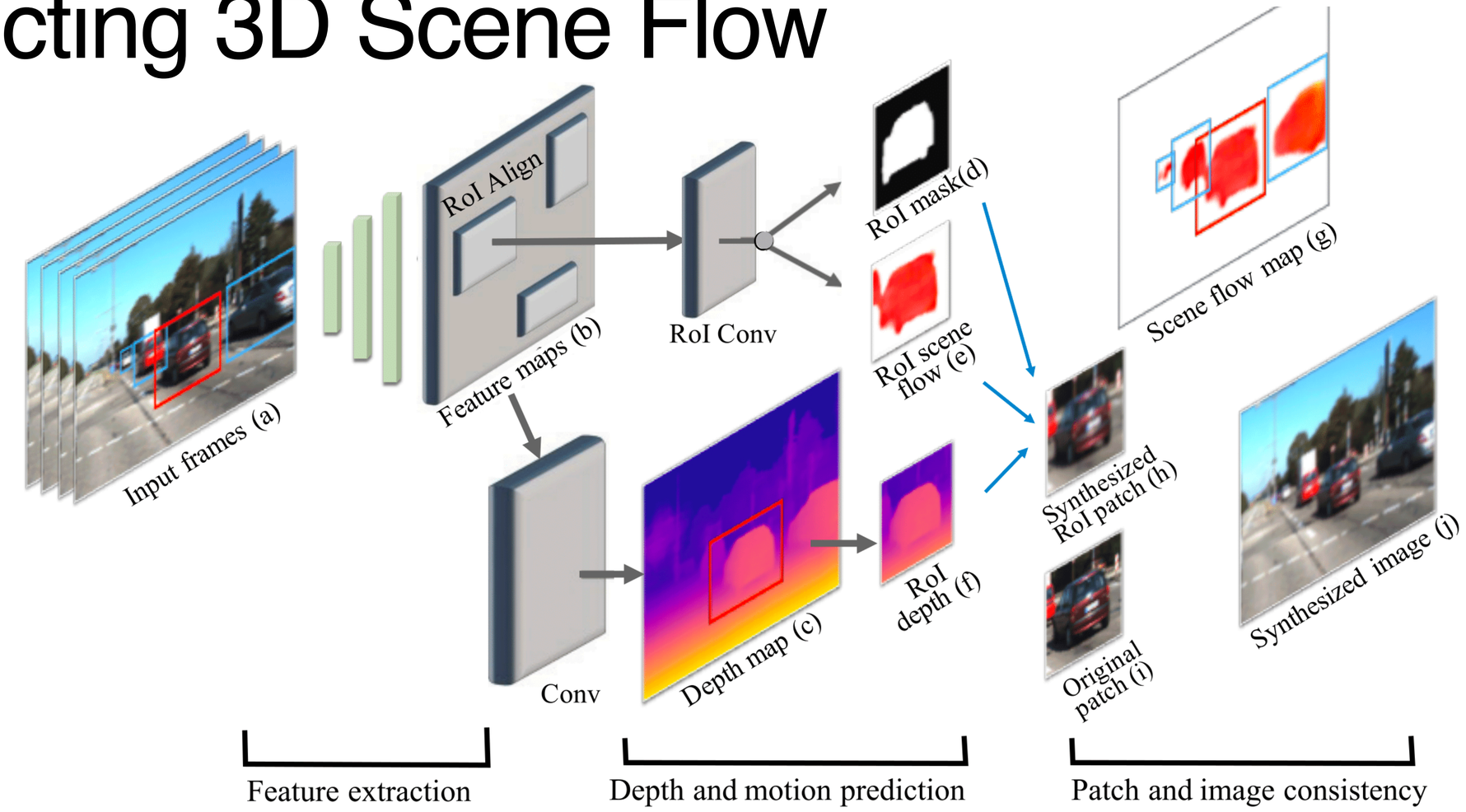


Image Courtesy: Learning Independent Object Motion from Unlabelled Stereoscopic Videos. Cao et. al.

Predicting 3D Scene Flow



Self-supervised learning of depth and scene flow

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

Predicting 3D Scene Flow

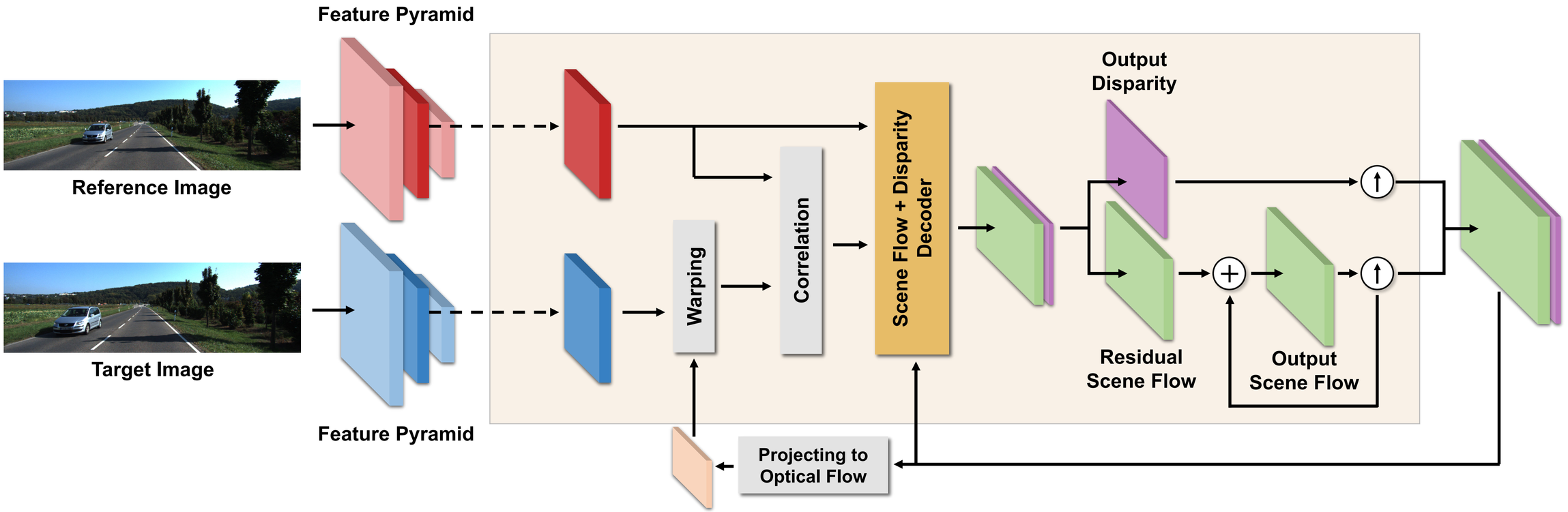
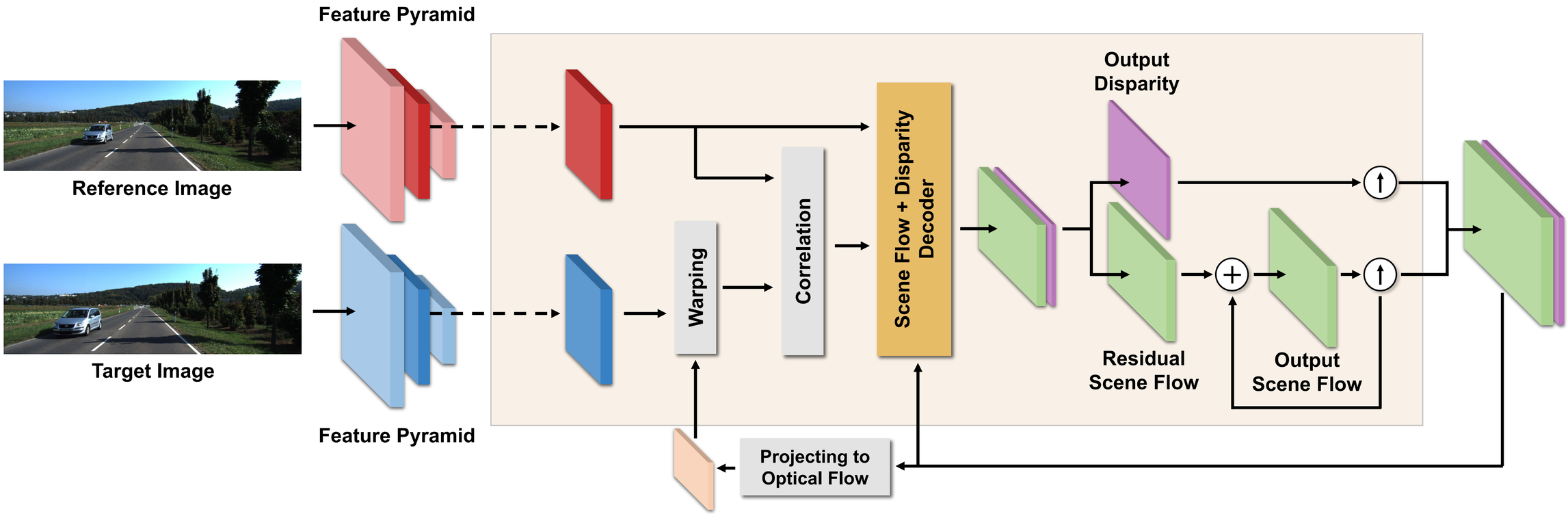


Image Courtesy: Self-Supervised Monocular Scene Flow Estimation. Hur et. al.

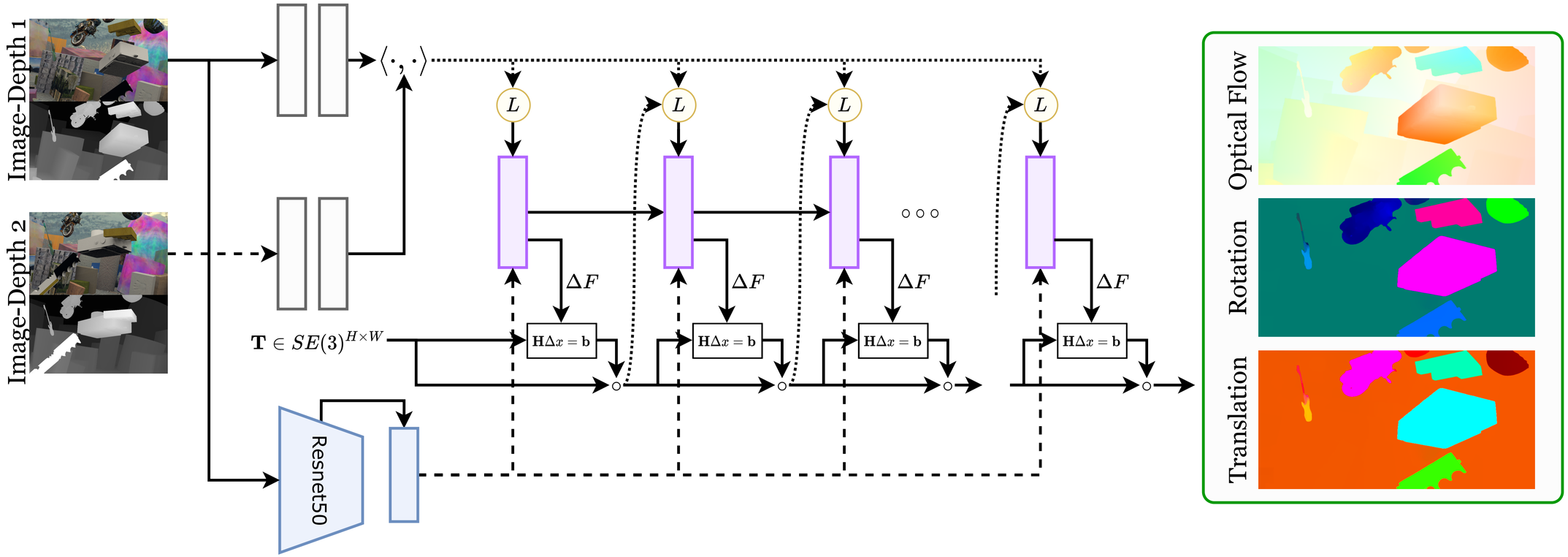
Predicting 3D Scene Flow



Self-supervised learning of depth and scene flow

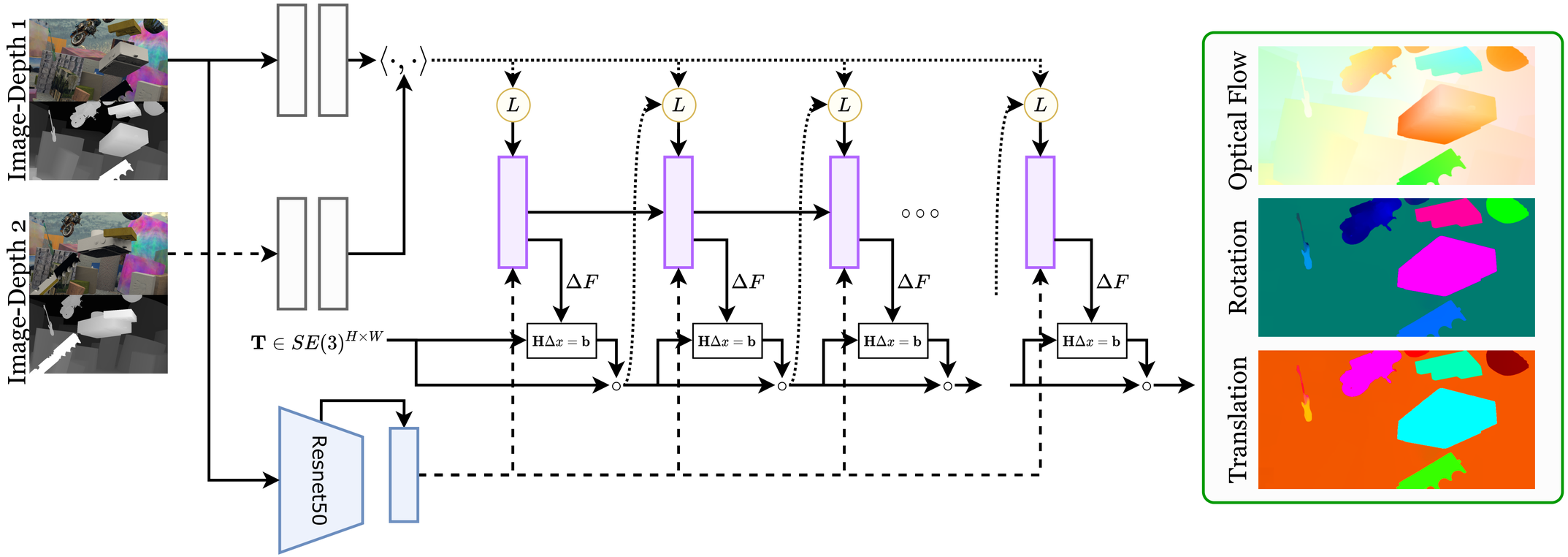
Image Courtesy: Self-Supervised Monocular Scene Flow Estimation. Hur et. al.

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

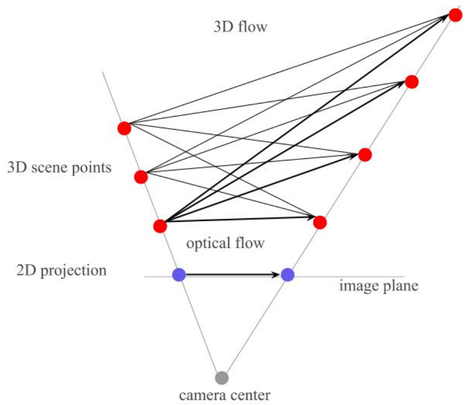
This allows straight-forward smoothness regularization!

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

3D Scene Flow: Some takeaways

3D Scene Flow: Some takeaways

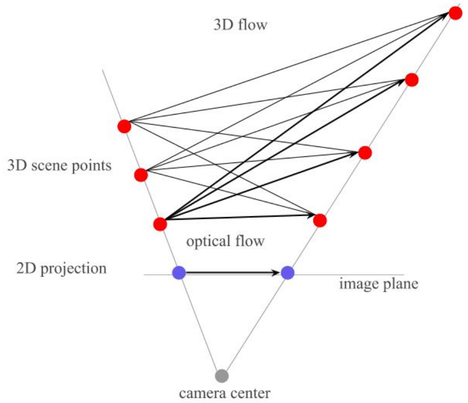
Scene flow allows predicting optical flow



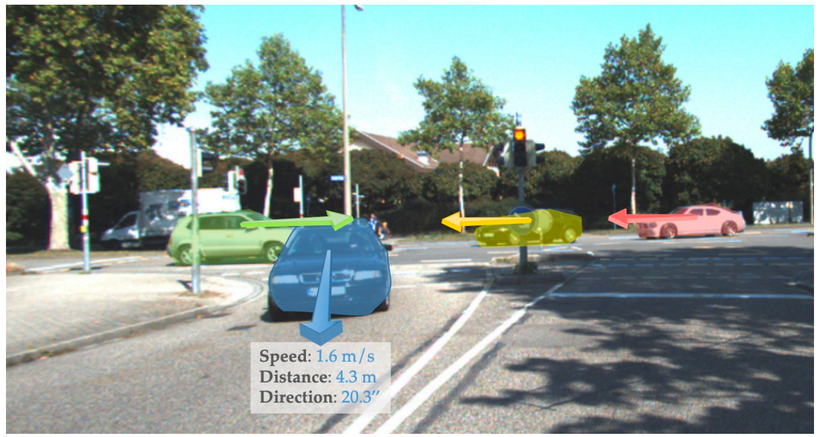
Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

3D Scene Flow: Some takeaways

Scene flow allows predicting optical flow



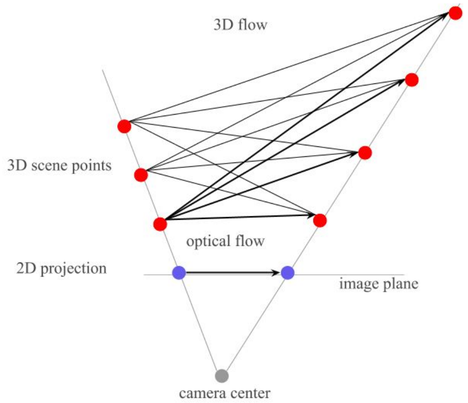
Typical motions are piecewise rigid



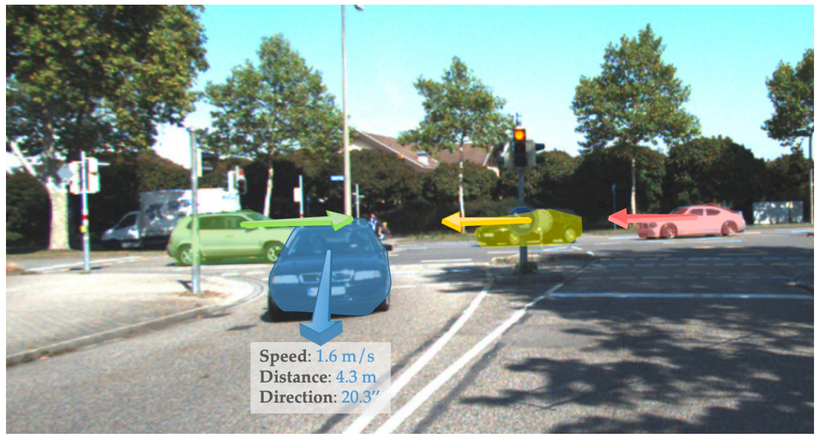
Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

3D Scene Flow: Some takeaways

Scene flow allows predicting optical flow



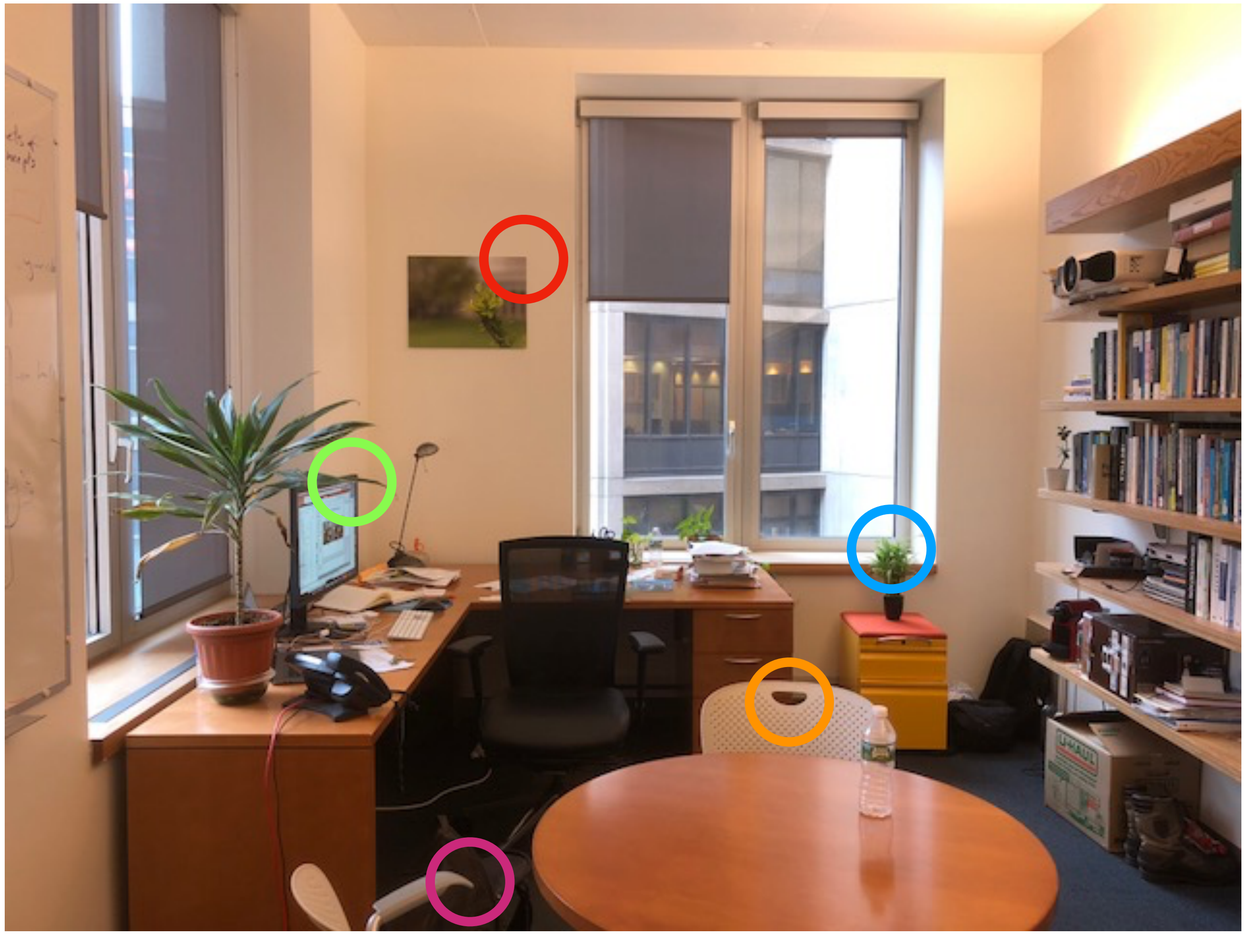
Typical motions are piecewise rigid

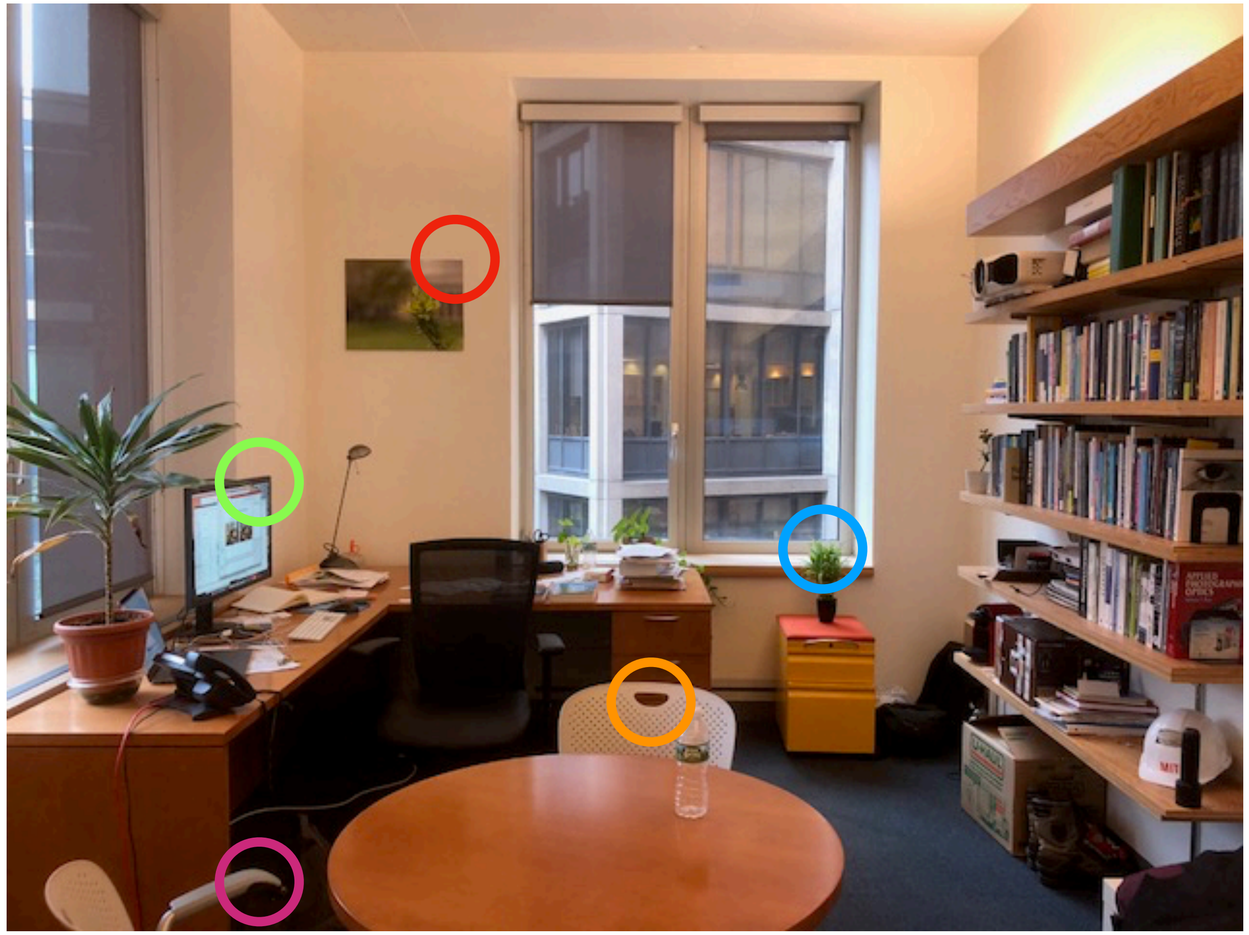


Possible parametrization as pointwise SE3 field (instead of pointwise translation)

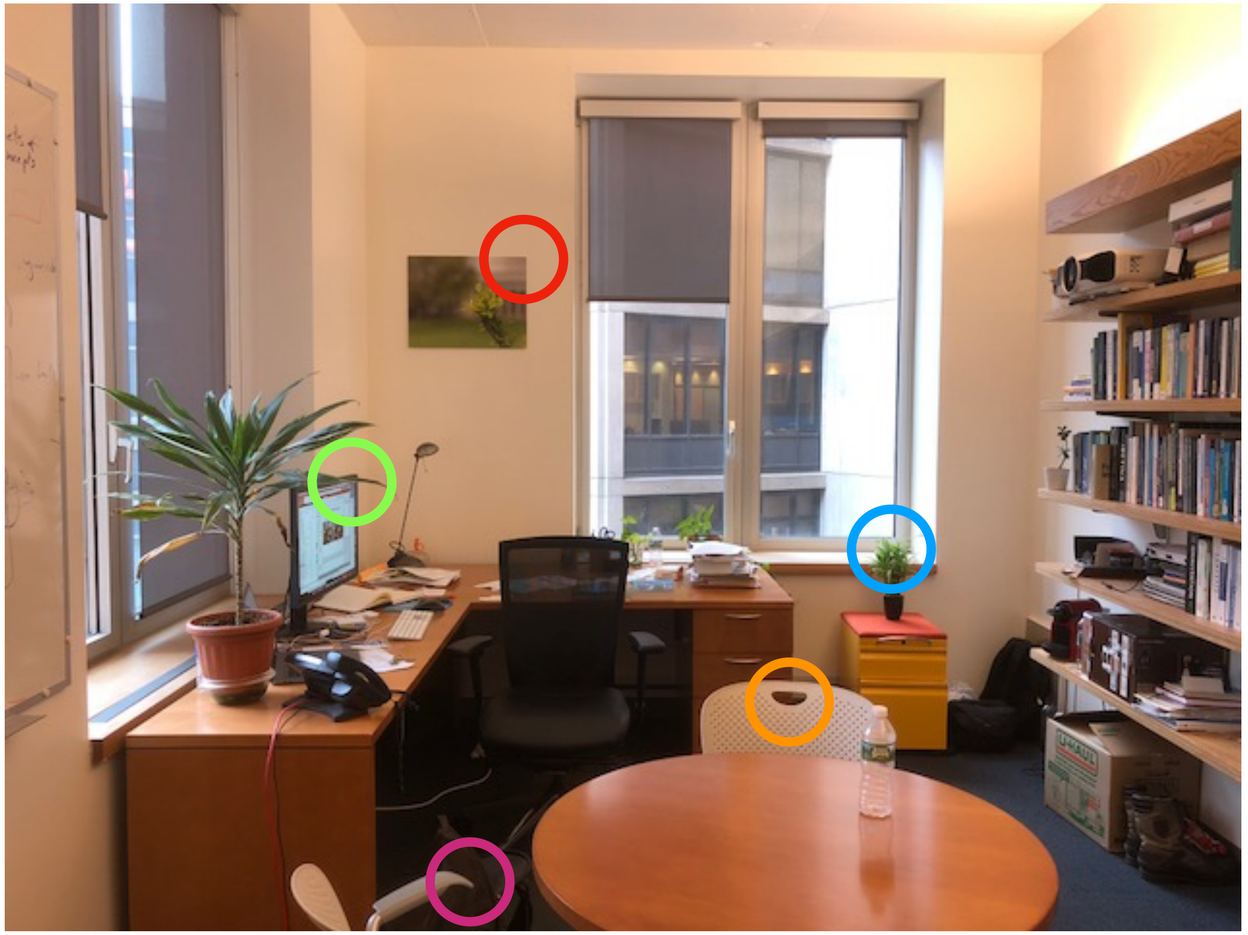
Slide curtesy of Shubham Tulsiani, CMU 16-889: Learning for 3D Vision

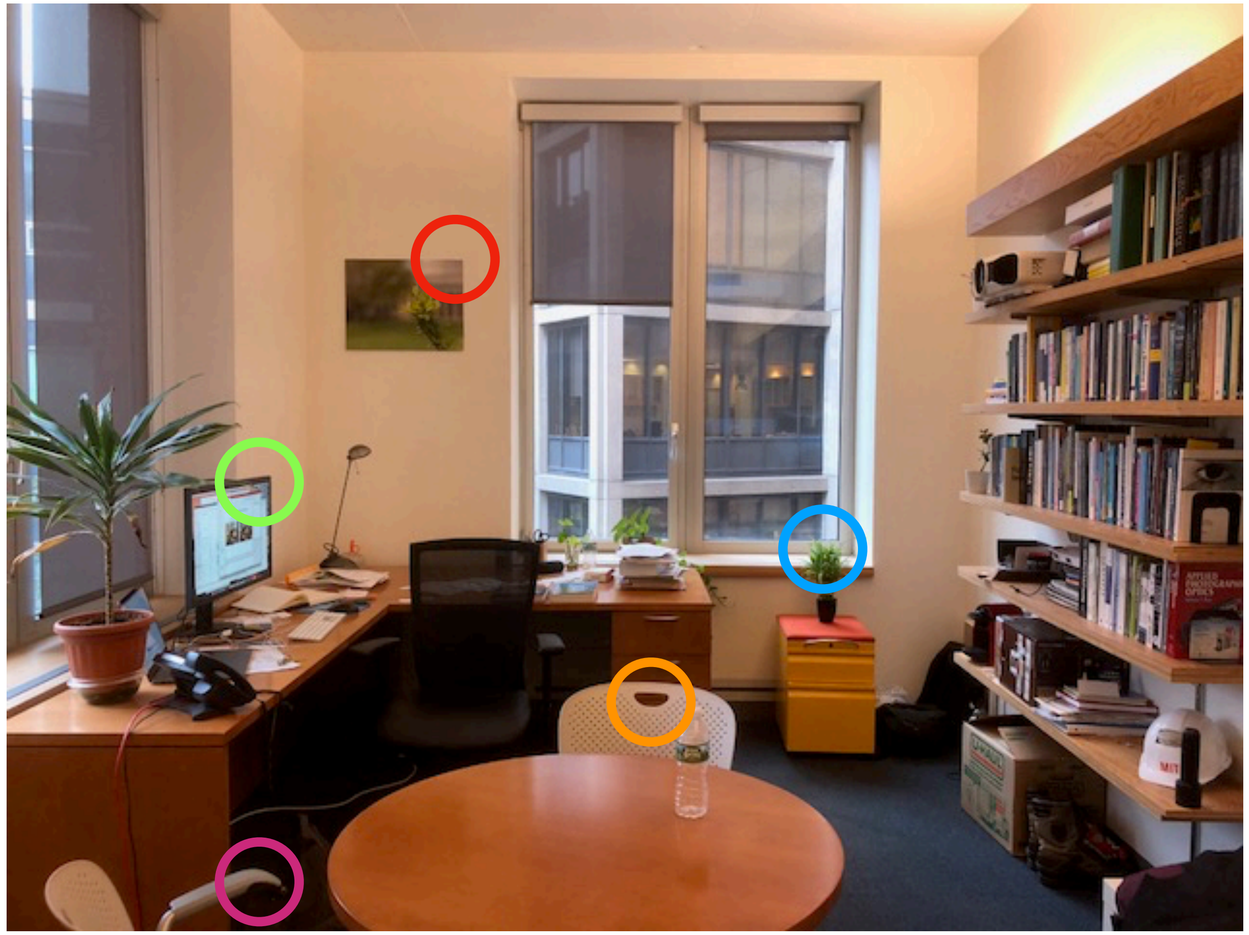
What if it’s not video, but multi-view images?





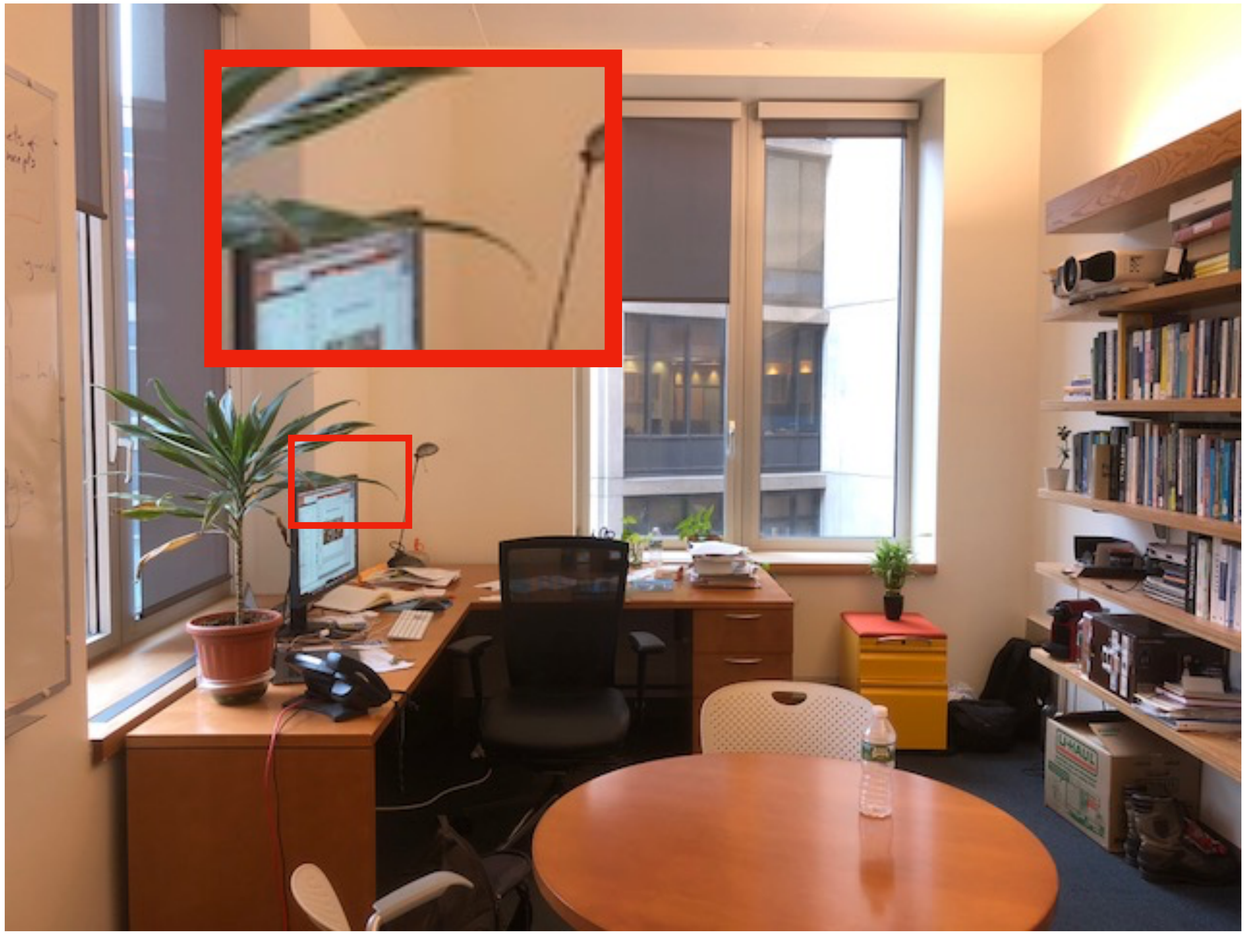
What if it’s not video, but multi-view images?



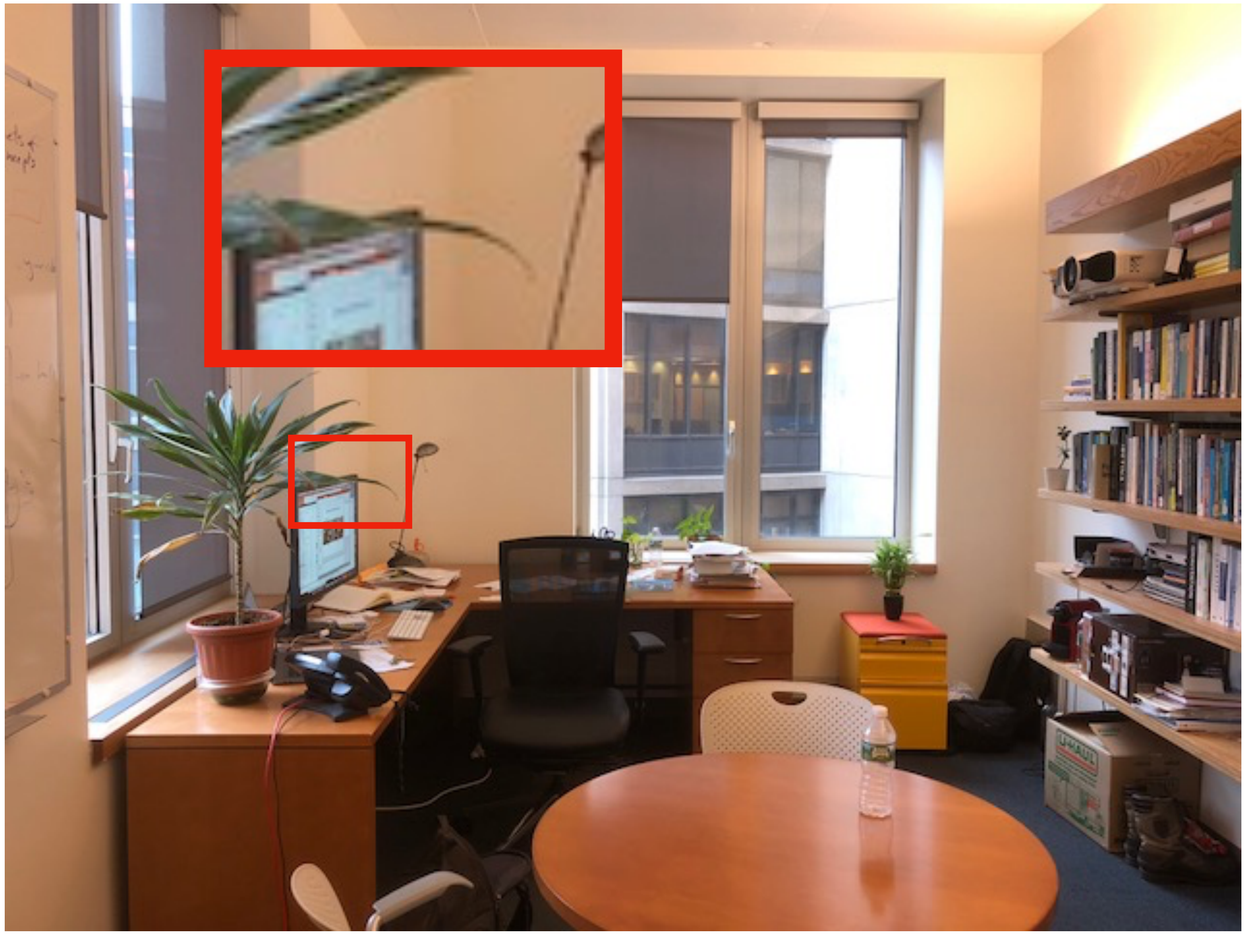


Correspondences aren’t *local* anymore.

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?

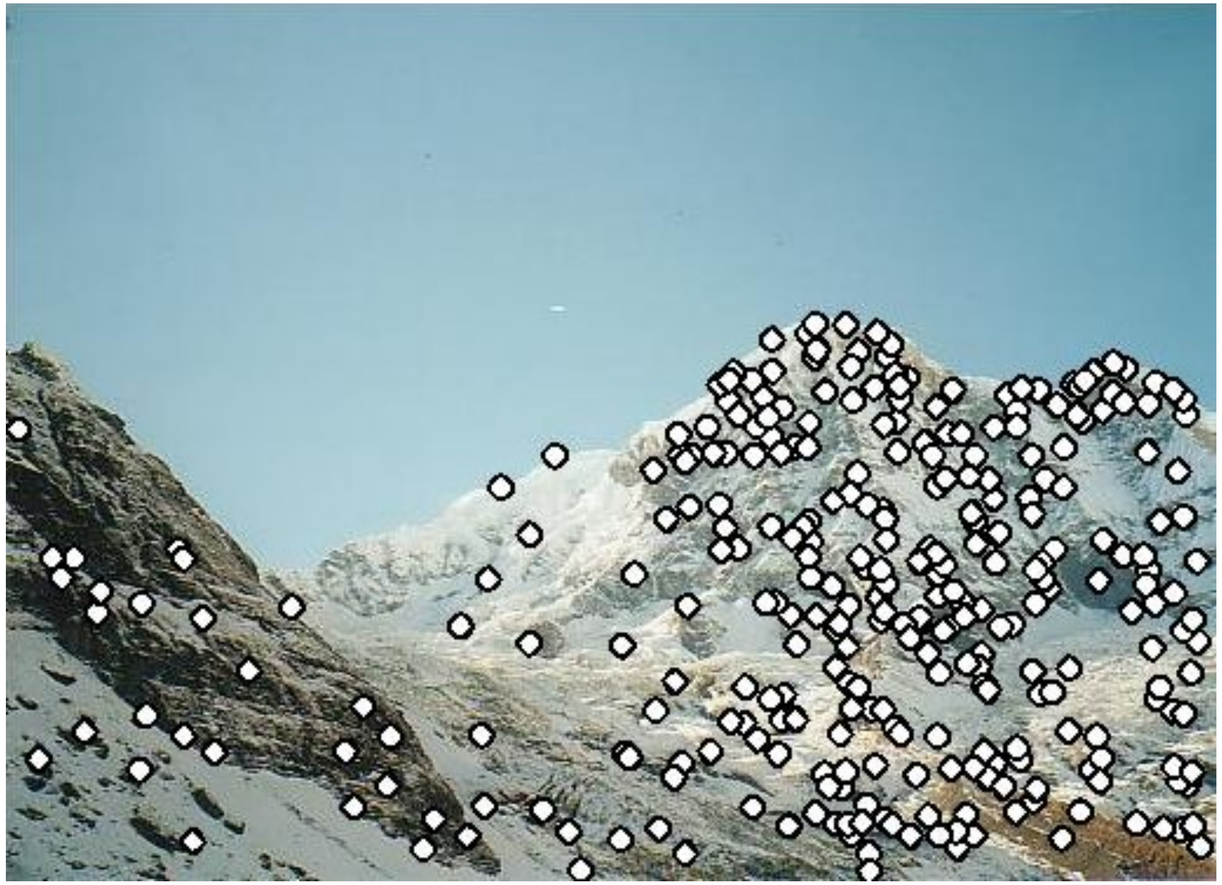
What if it’s not video, but multi-view images?

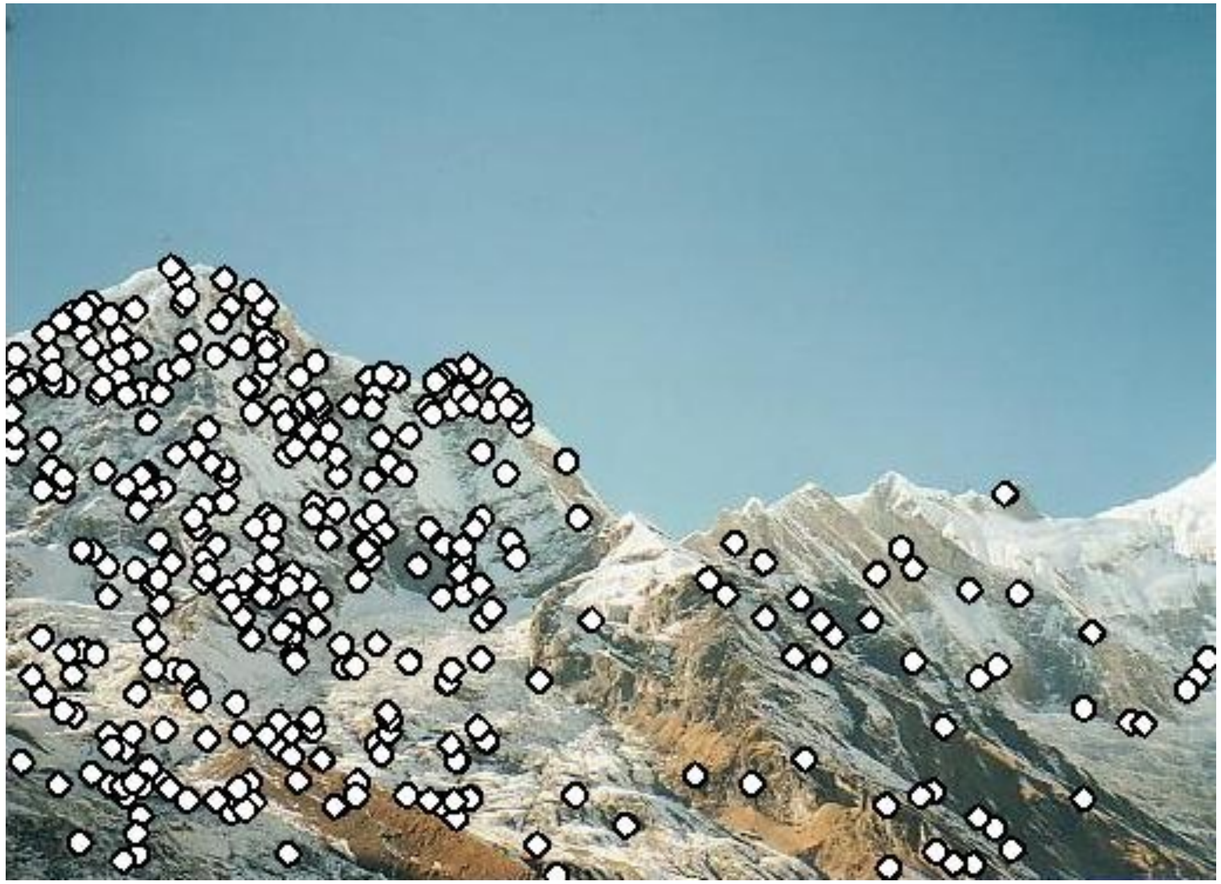
We want to be robust to view-dependent effects and lighting changes.

Feature Detection, Description & Matching

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

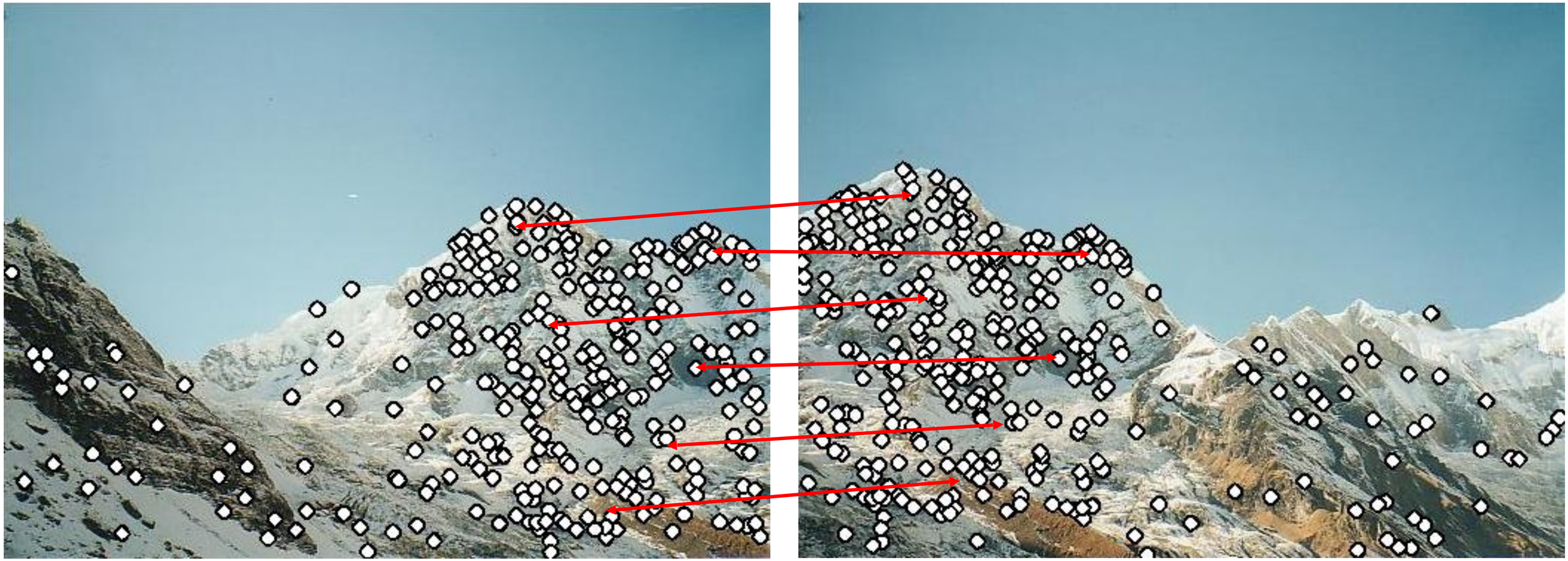
Feature Detection, Description & Matching





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

Feature Detection, Description & Matching

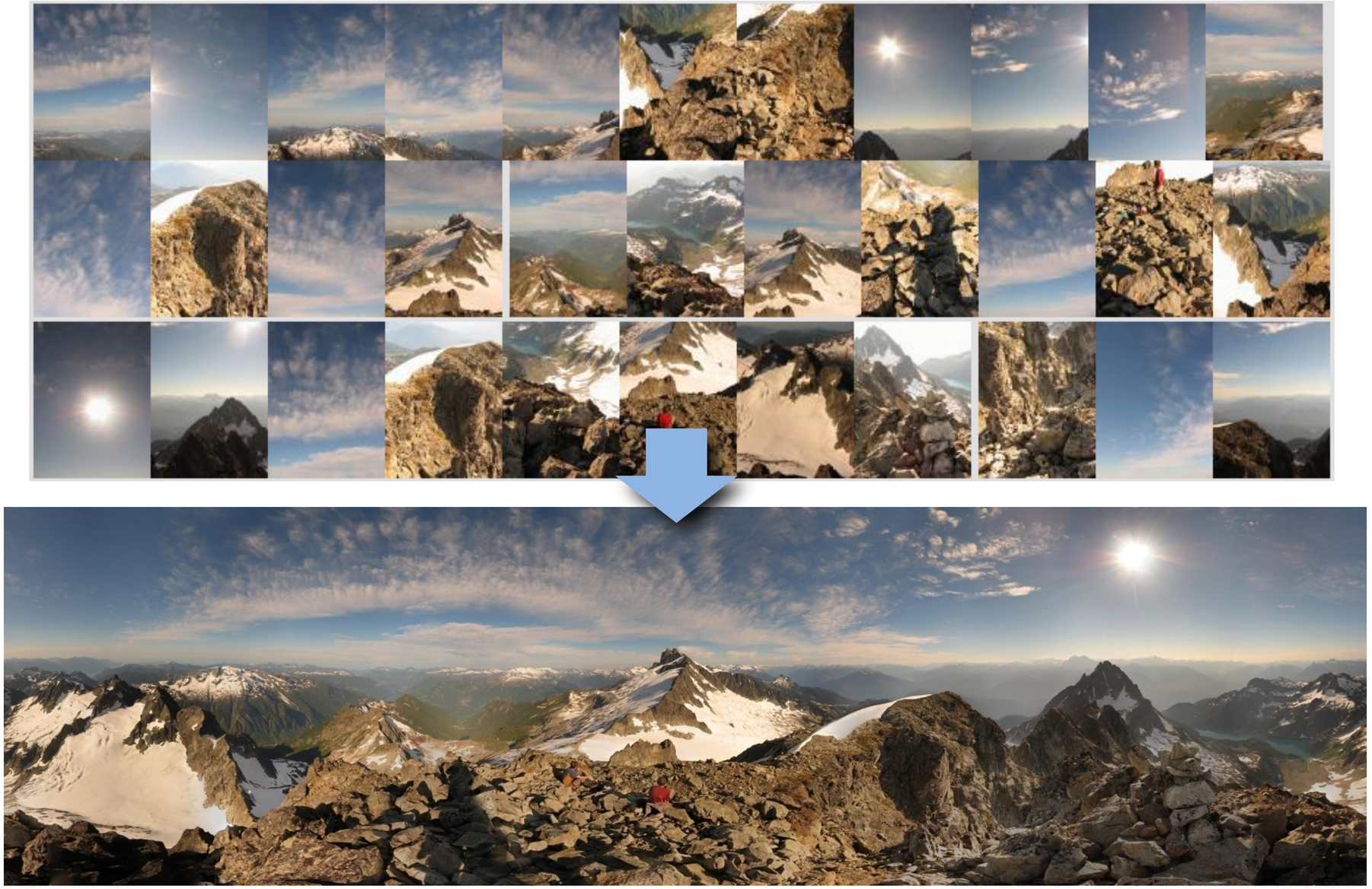


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Application 1: Panorama Stitching

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

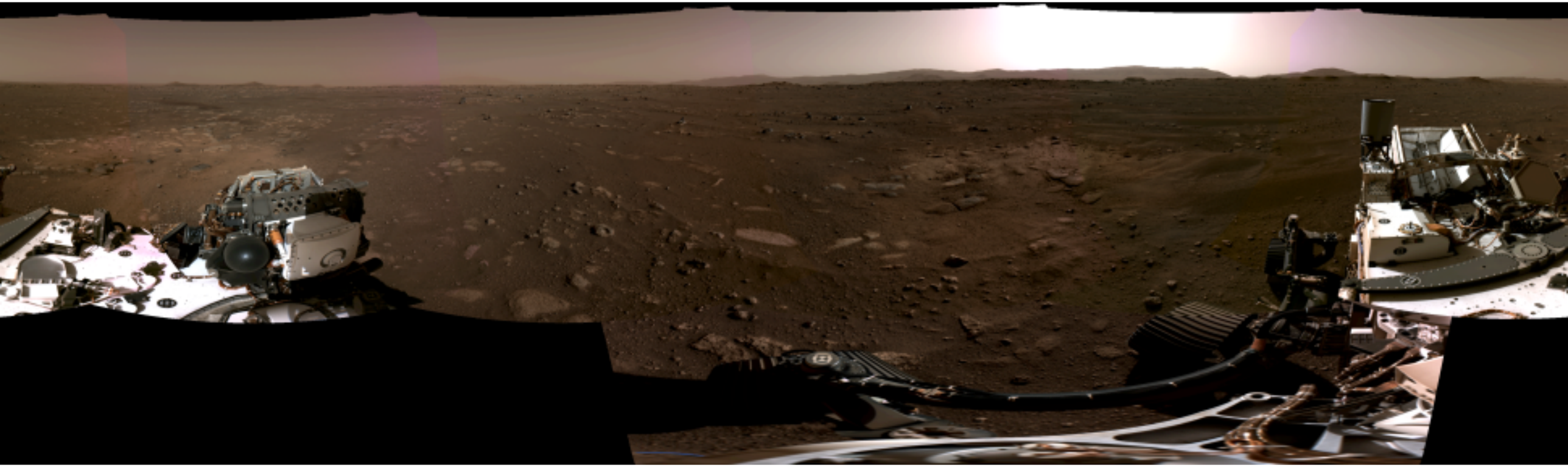
More Automatic panoramas



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

Credit: Matt Brown

Panorama stitching

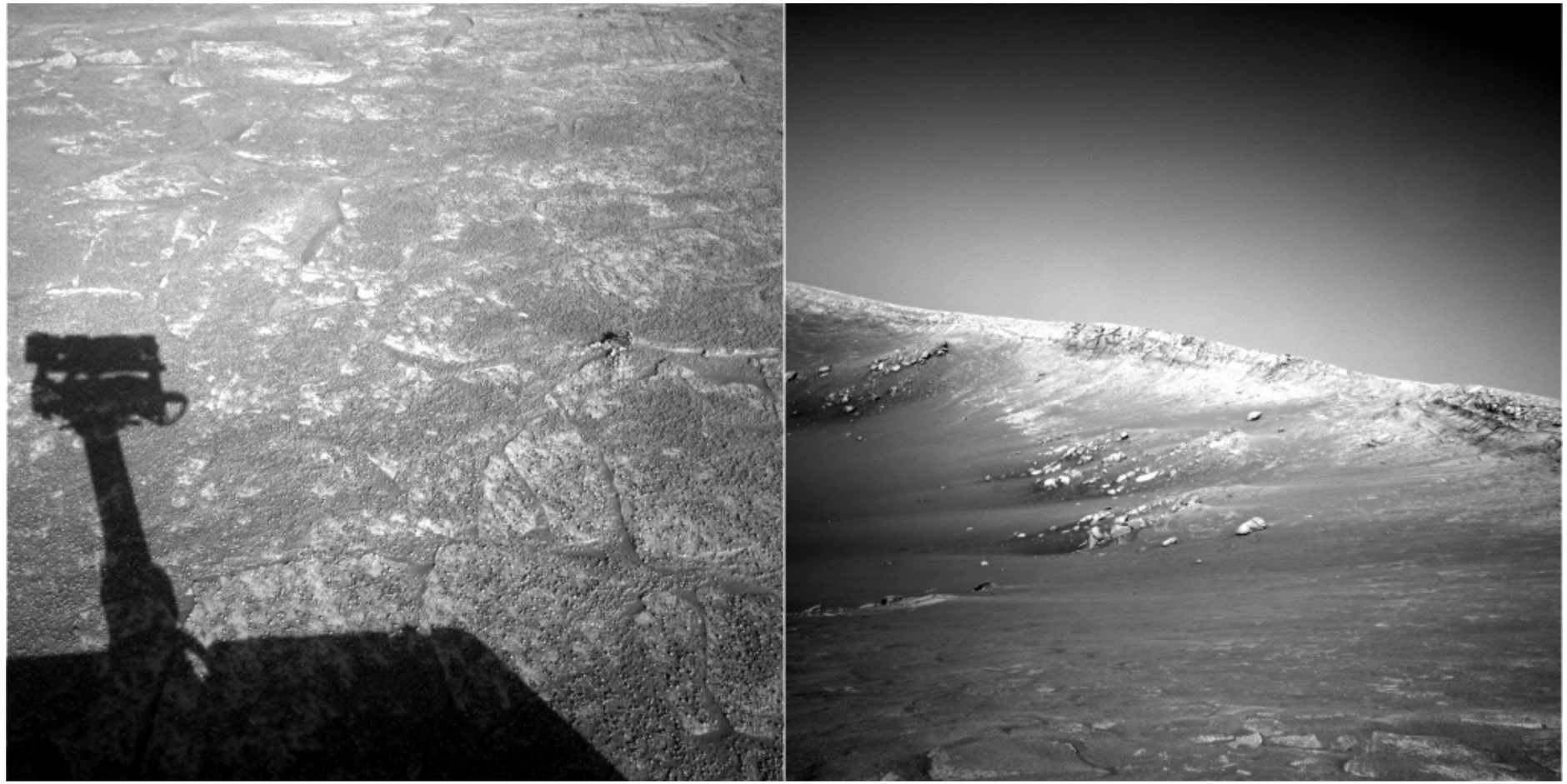


Panorama captured by Perseverence Rover, Feb. 20, 2021

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

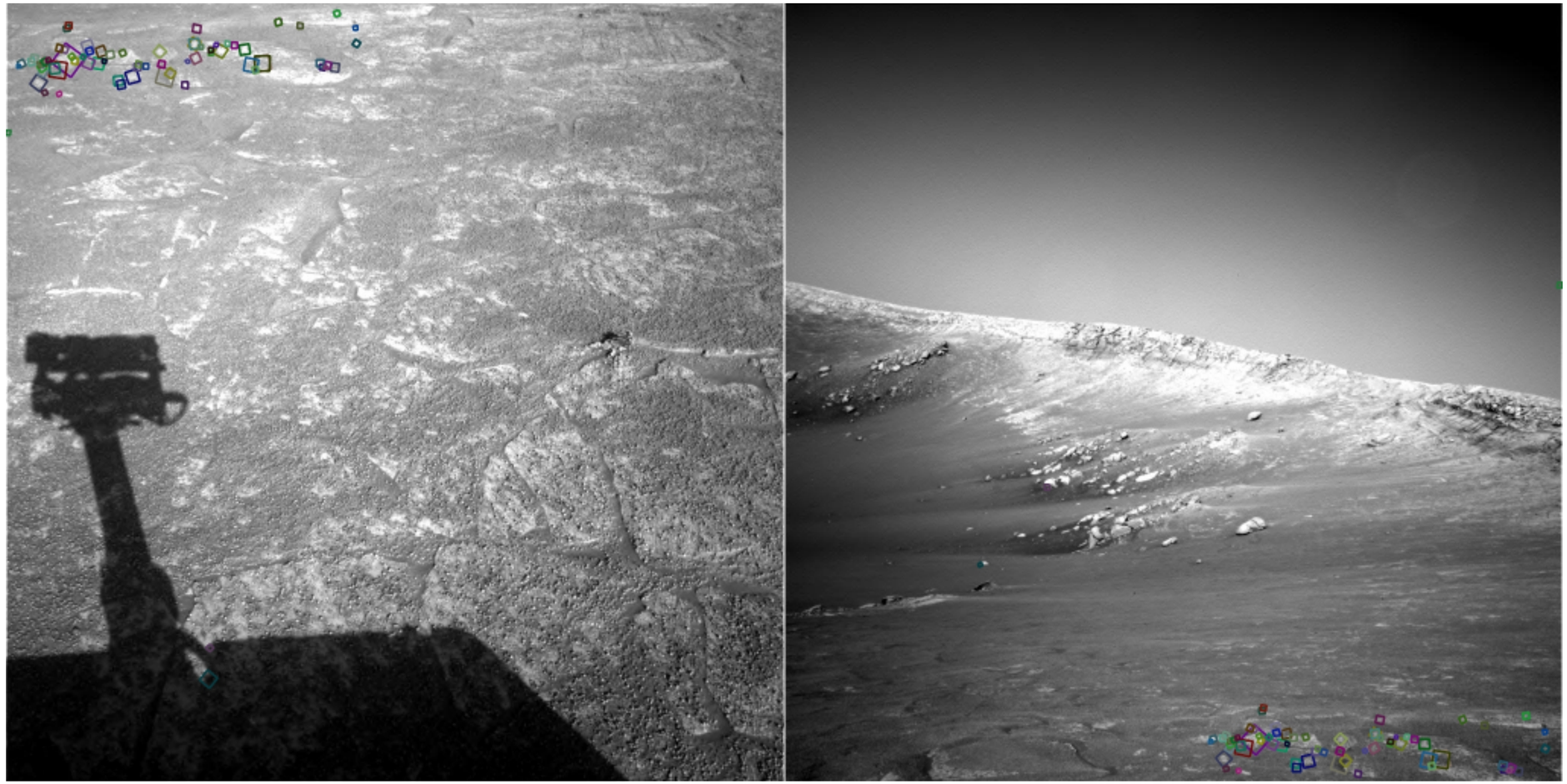
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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…)



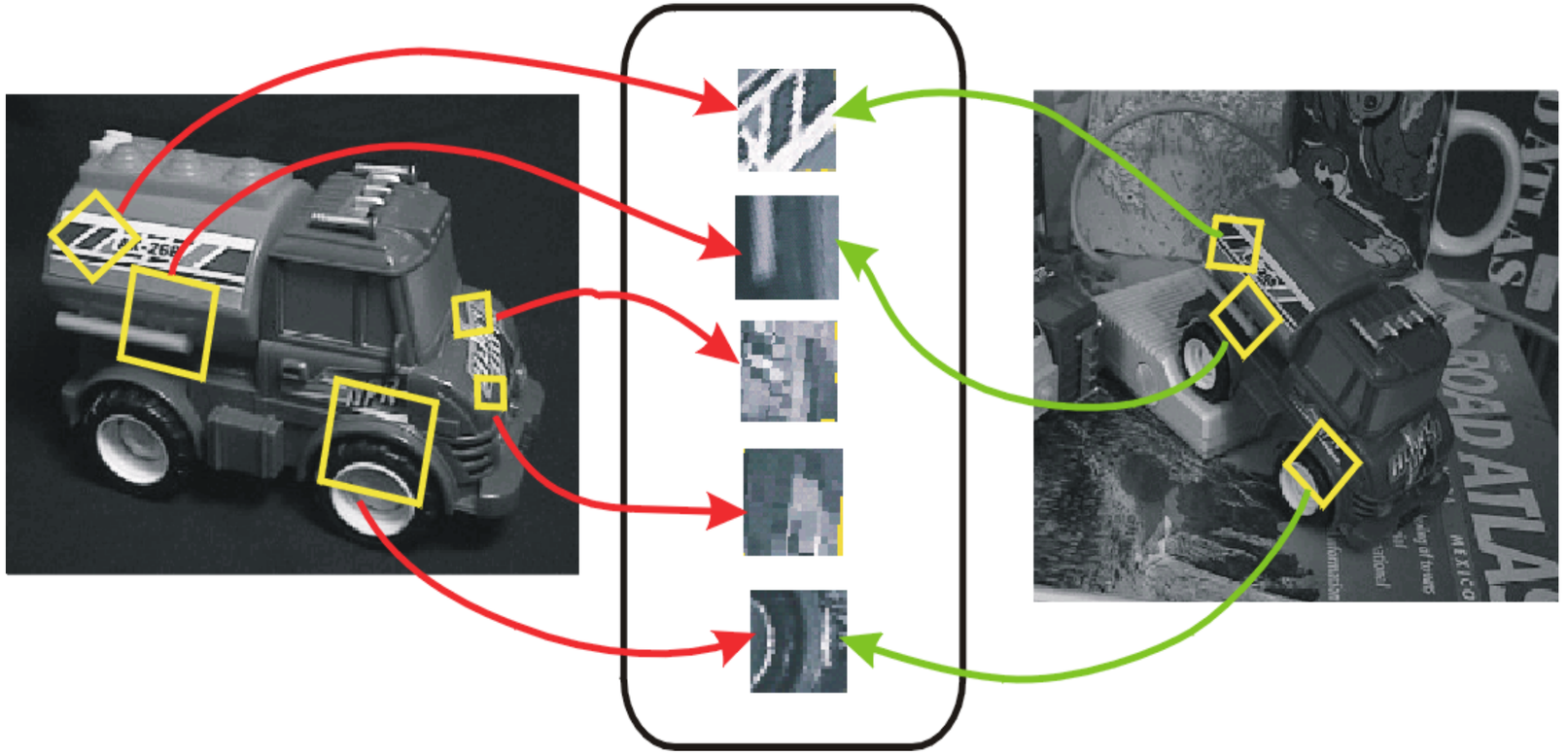
NASA Mars Rover images
with SIFT feature matches

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, …

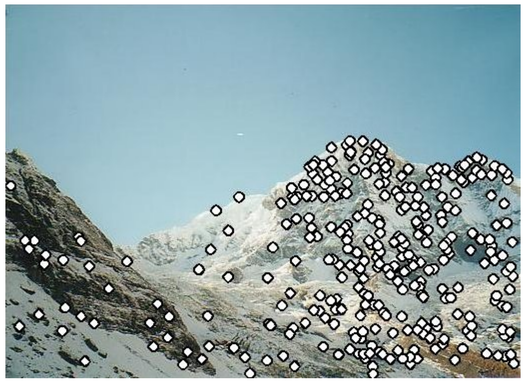


Feature Descriptors

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

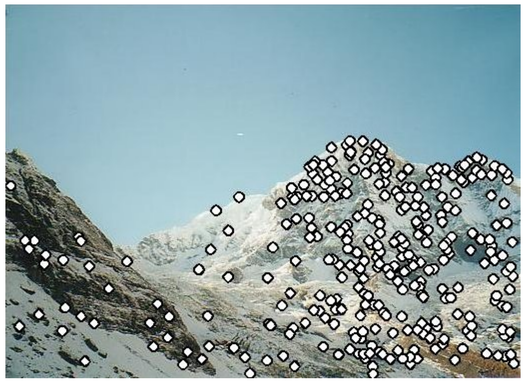


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

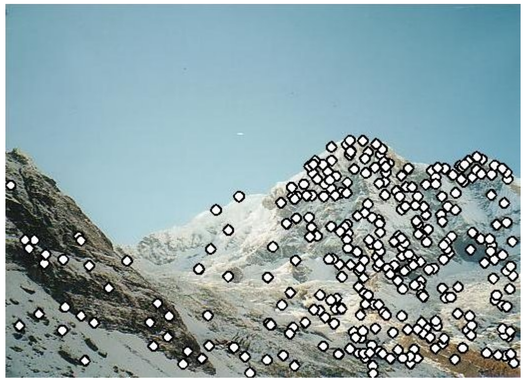
Credit: Kristen Grauman

Local features: main components

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



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

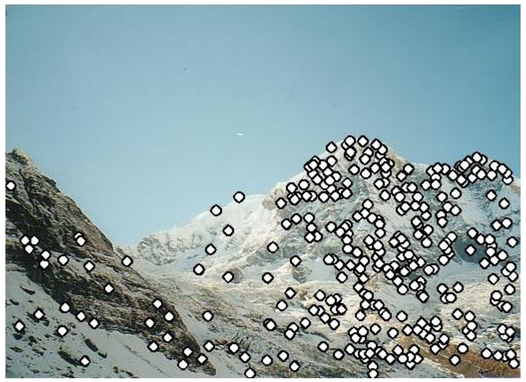


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

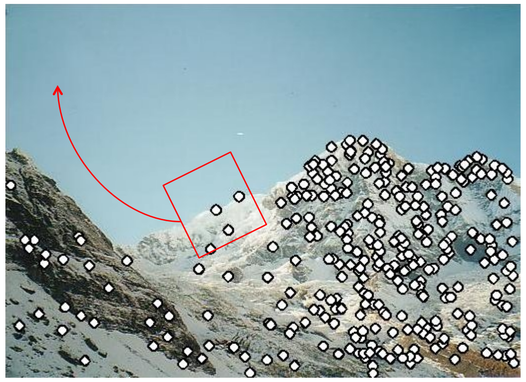
Credit: Kristen Grauman

Local features: main components

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



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

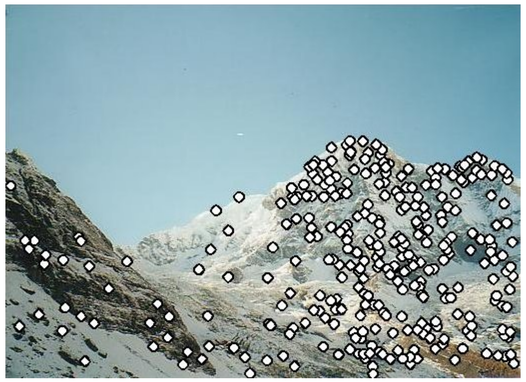


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

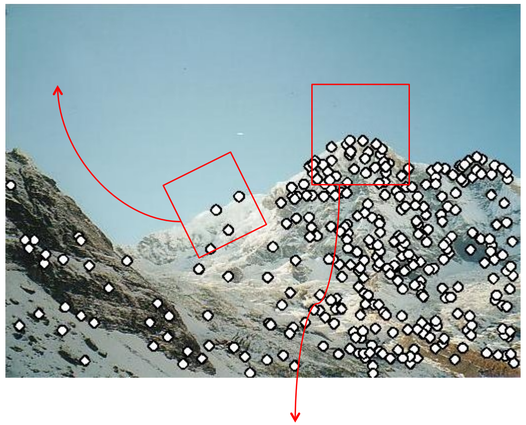
Credit: Kristen Grauman

Local features: main components

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



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

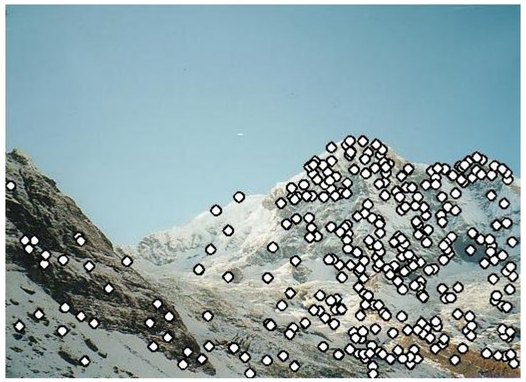


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

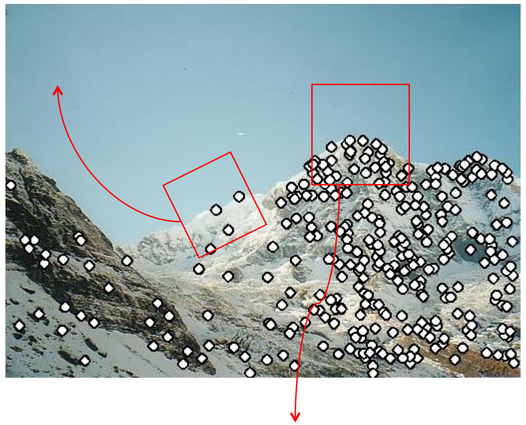
Credit: Kristen Grauman

Local features: main components

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



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



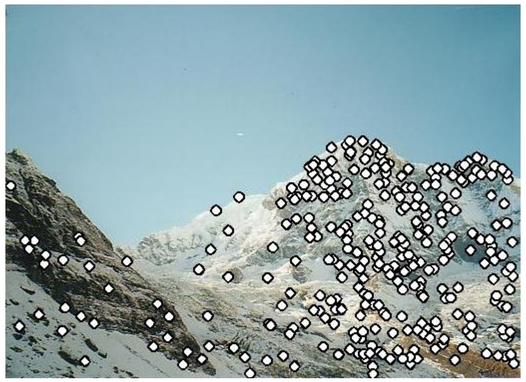
3) **Matching**: Determine correspondence between descriptors in two views

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

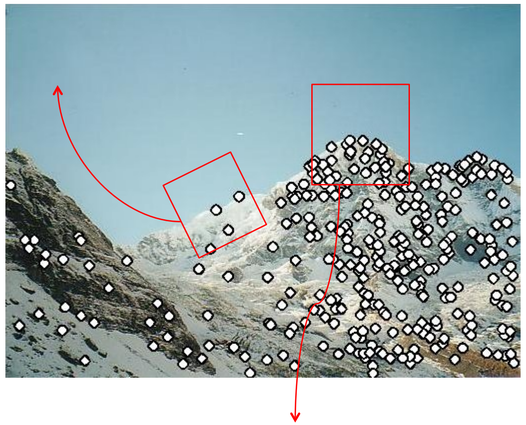
Credit: Kristen Grauman

Local features: main components

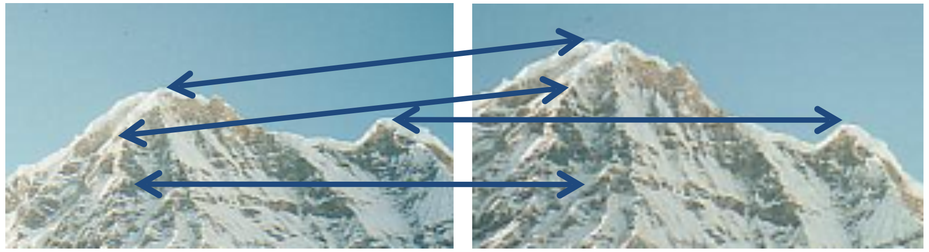
1) **Detection**: Identify the interest points



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



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

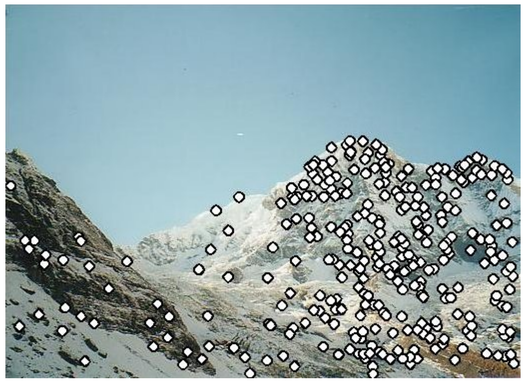


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Credit: Kristen Grauman

Local features: main components

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

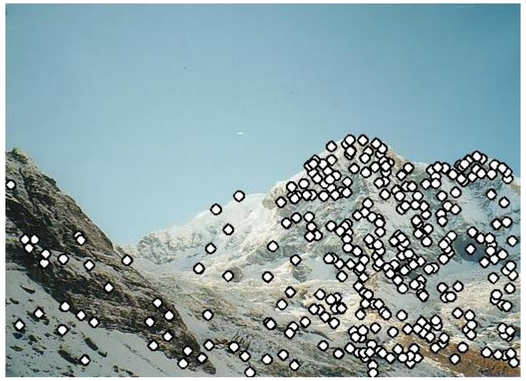


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

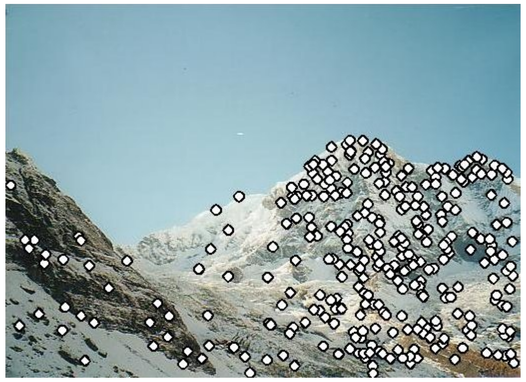
Credit: Kristen Grauman

Local features: main components

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



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

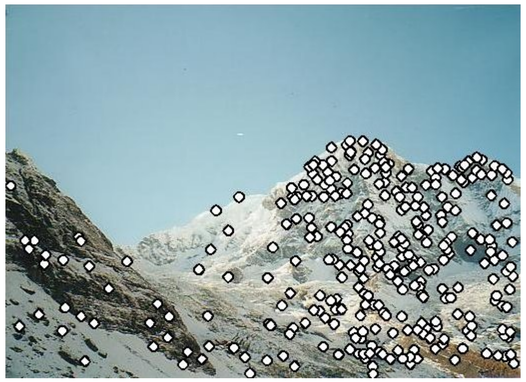


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

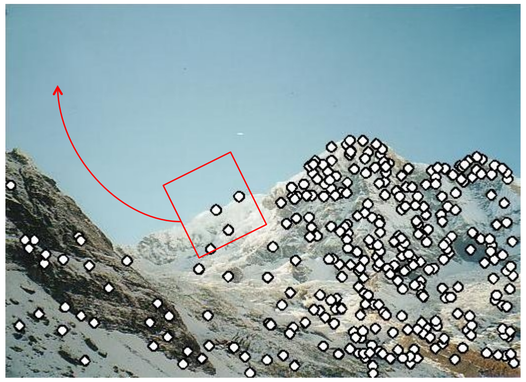
Credit: Kristen Grauman

Local features: main components

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



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

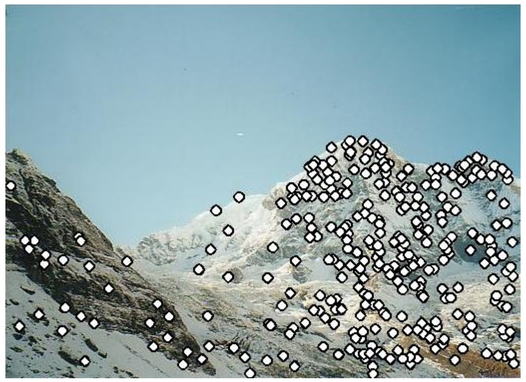


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

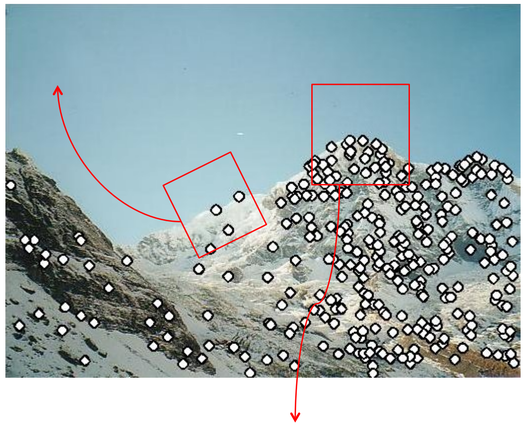
Credit: Kristen Grauman

Local features: main components

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



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

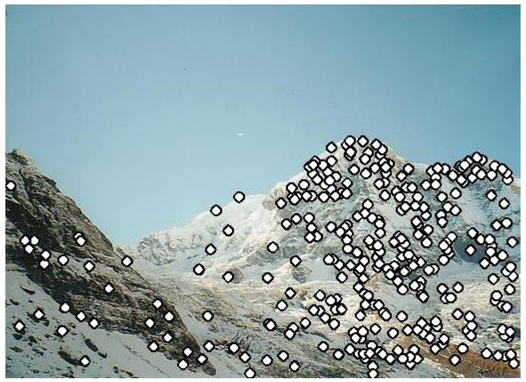


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

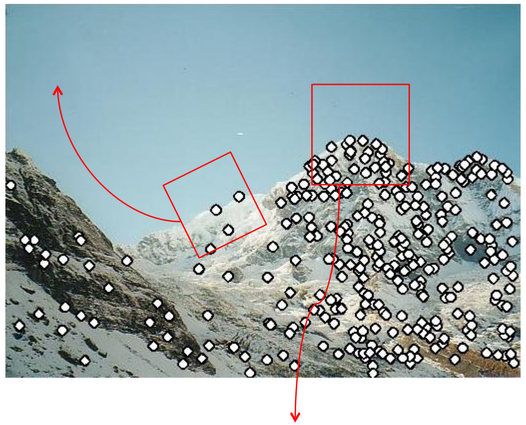
Credit: Kristen Grauman

Local features: main components

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



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



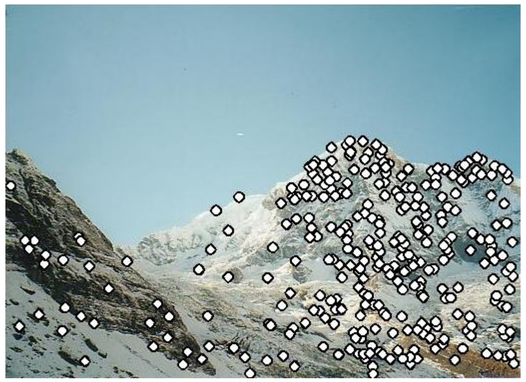
3) **Matching**: Determine correspondence between descriptors in two views

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

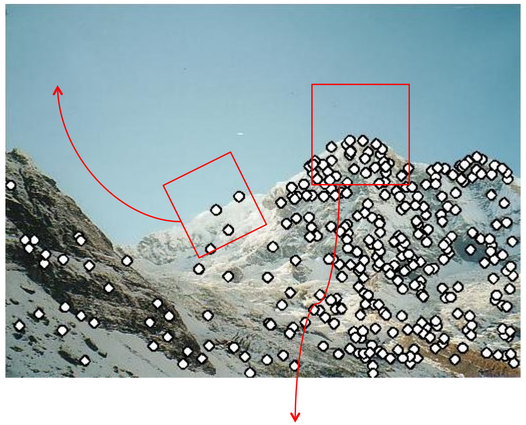
Credit: Kristen Grauman

Local features: main components

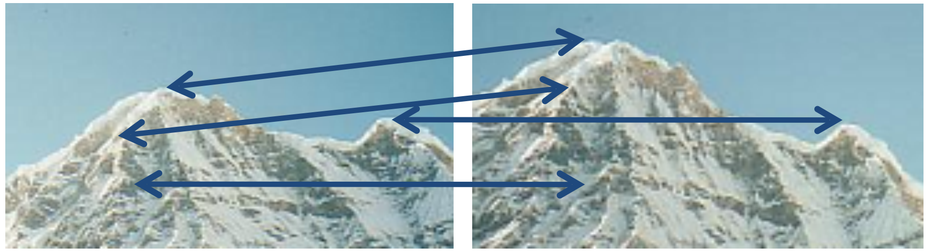
1) **Detection: Identify the interest points**



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

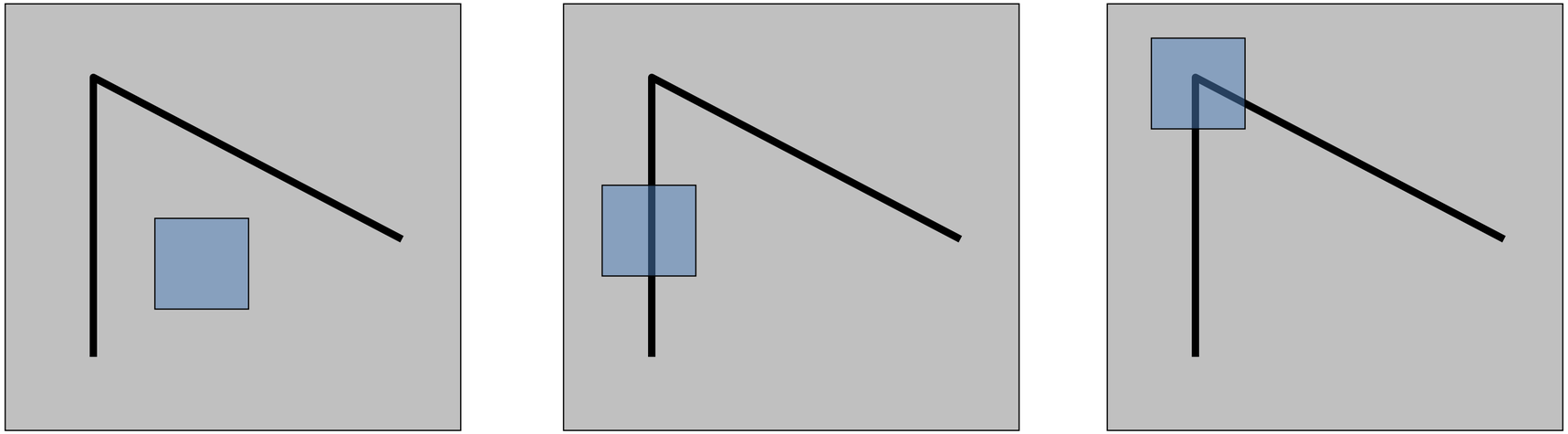
How to define “unusual”?

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?

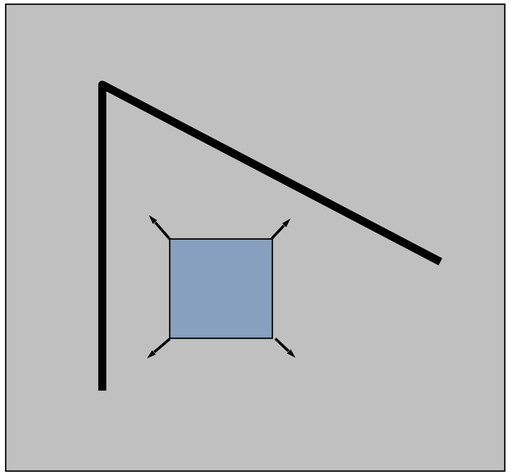


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

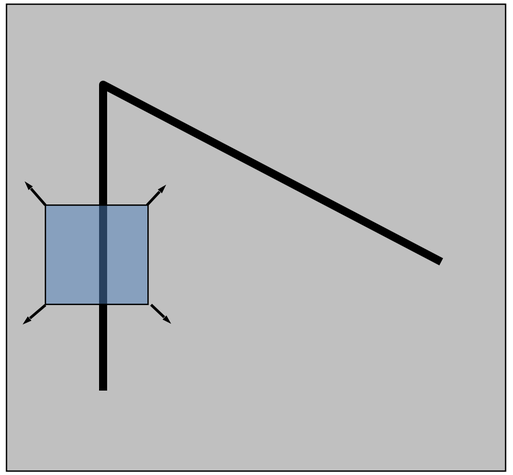
Credit: S. Seitz, D. Frolova, D. Simakov

Local measures of uniqueness

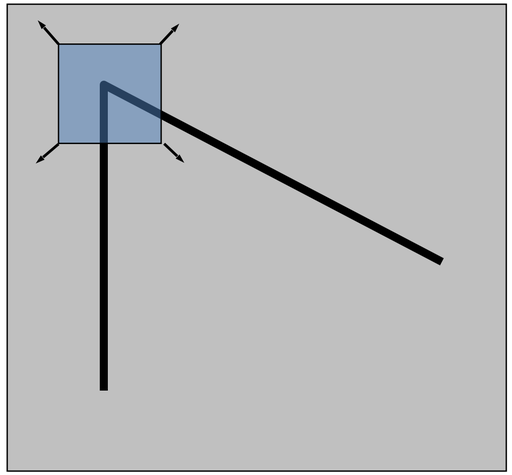
- How does the window change when you shift it?
- Shifting the window in any direction causes a big change



“flat” region:
no change in all directions



“edge”:
no change along the edge direction



“corner”:
significant change in all directions

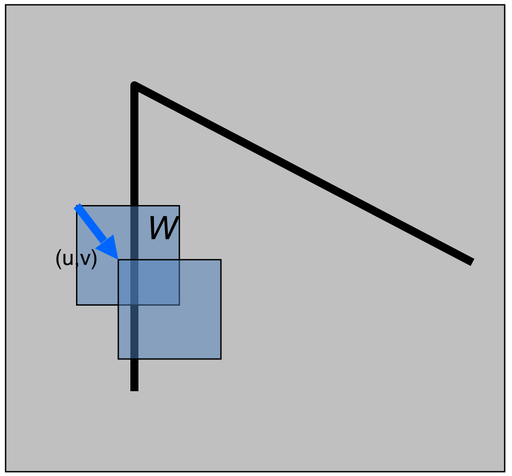
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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)*:



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

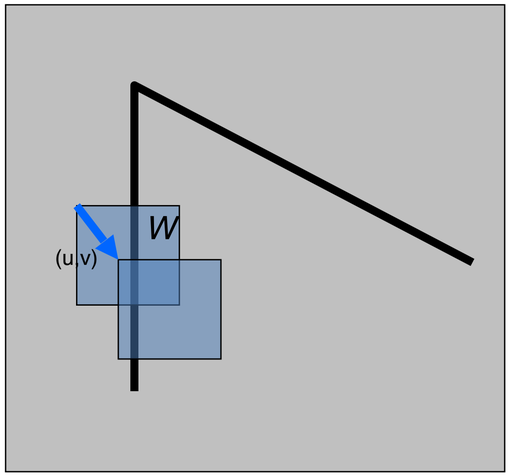
[Taylor expansion]

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)*:



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

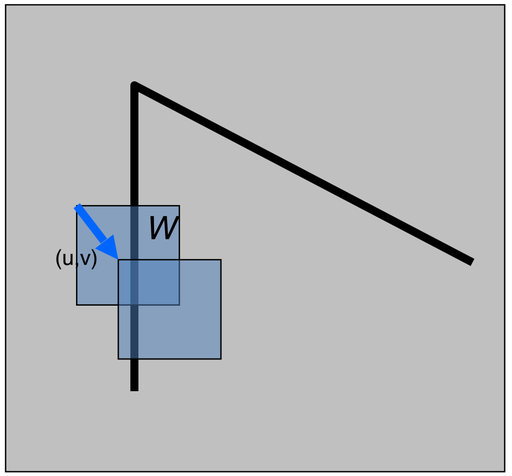
$$\text{[Taylor expansion]} \approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2$$

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

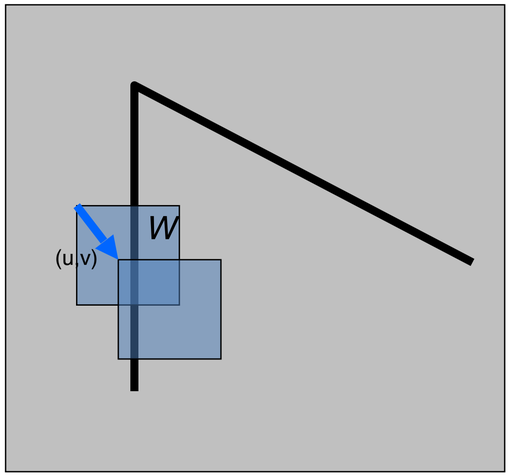
$$\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

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)*:

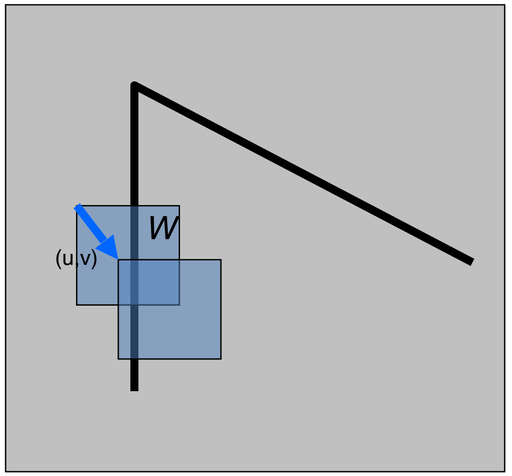


$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

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

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)*:

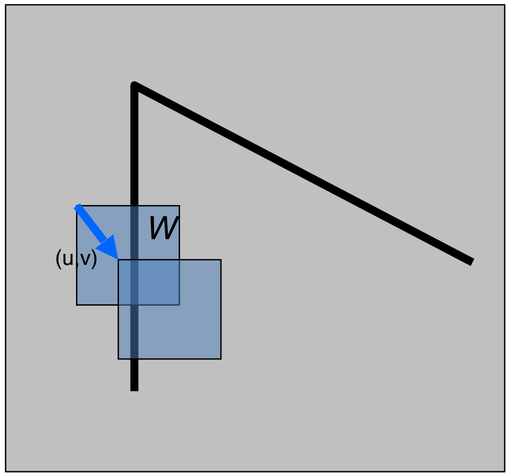
$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

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

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

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

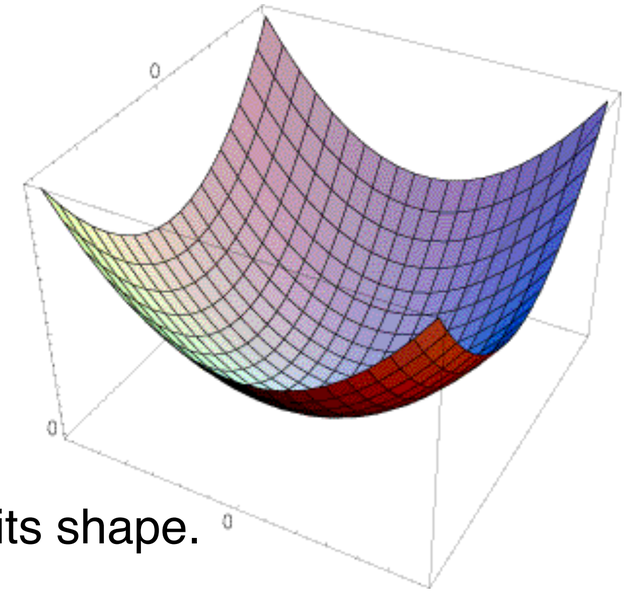
$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

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

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

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

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



Let's try to understand its shape.

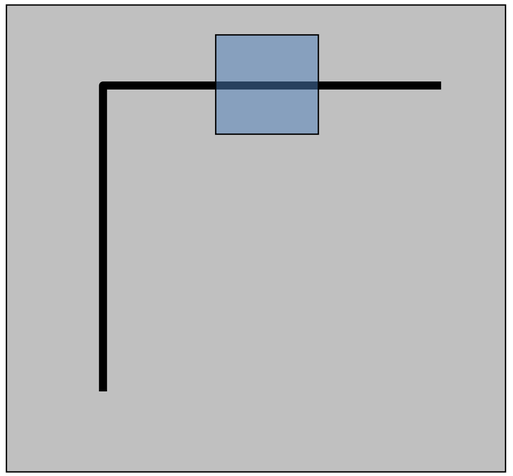
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

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

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



Horizontal edge: $I_x = 0$

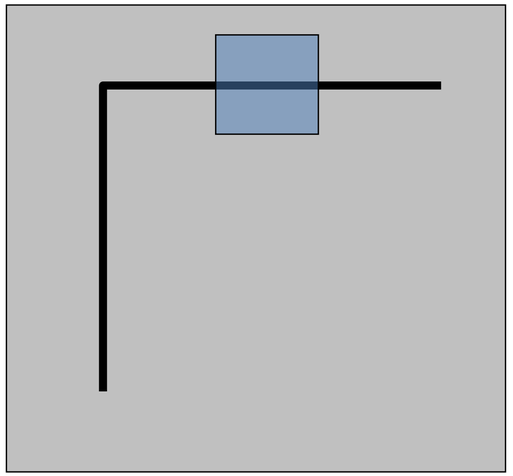
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

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

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



Horizontal edge: $I_x = 0$

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

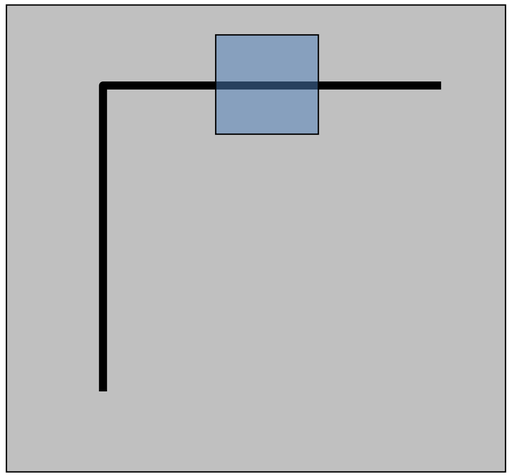
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

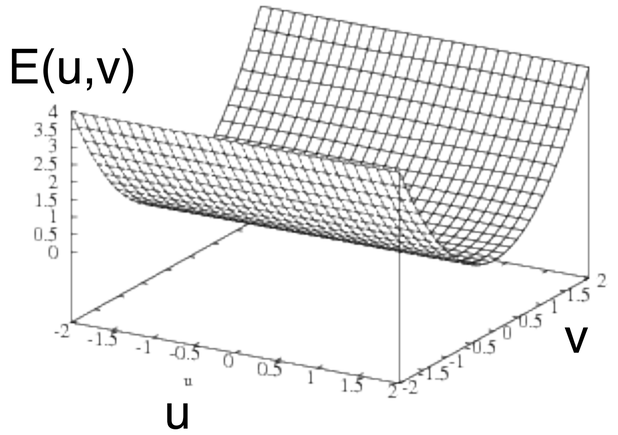
$$B = \sum_{(x,y)\in W} I_x I_y$$

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



Horizontal edge: $I_x = 0$

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



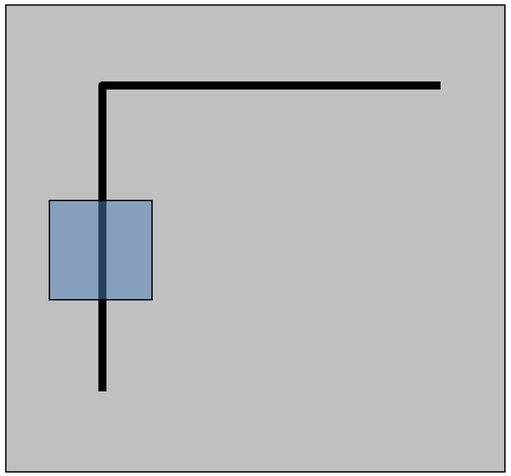
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

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

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



Vertical edge: $I_y = 0$

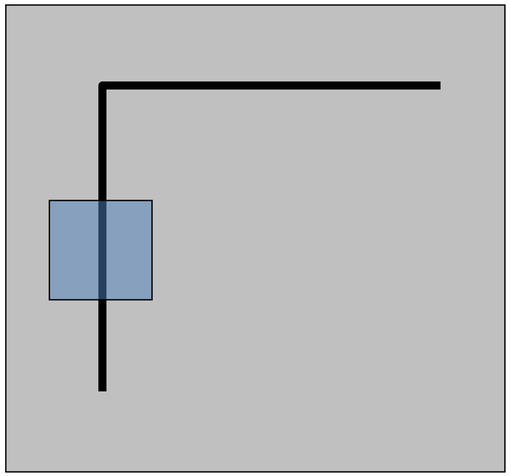
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

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

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



Vertical edge: $I_y = 0$

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

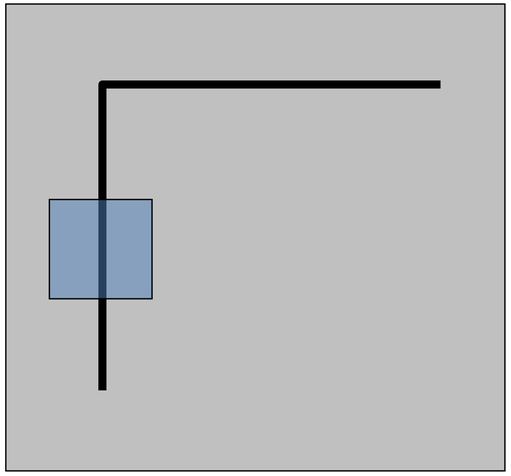
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

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

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

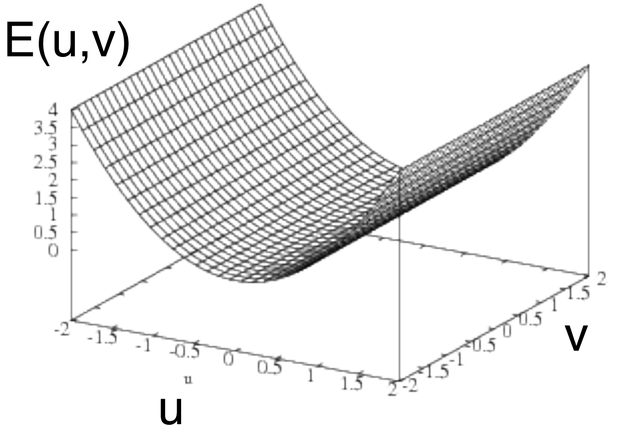
$$C = \sum_{(x,y)\in W} I_y^2$$



Vertical edge: $I_y = 0$

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

E(u,v)

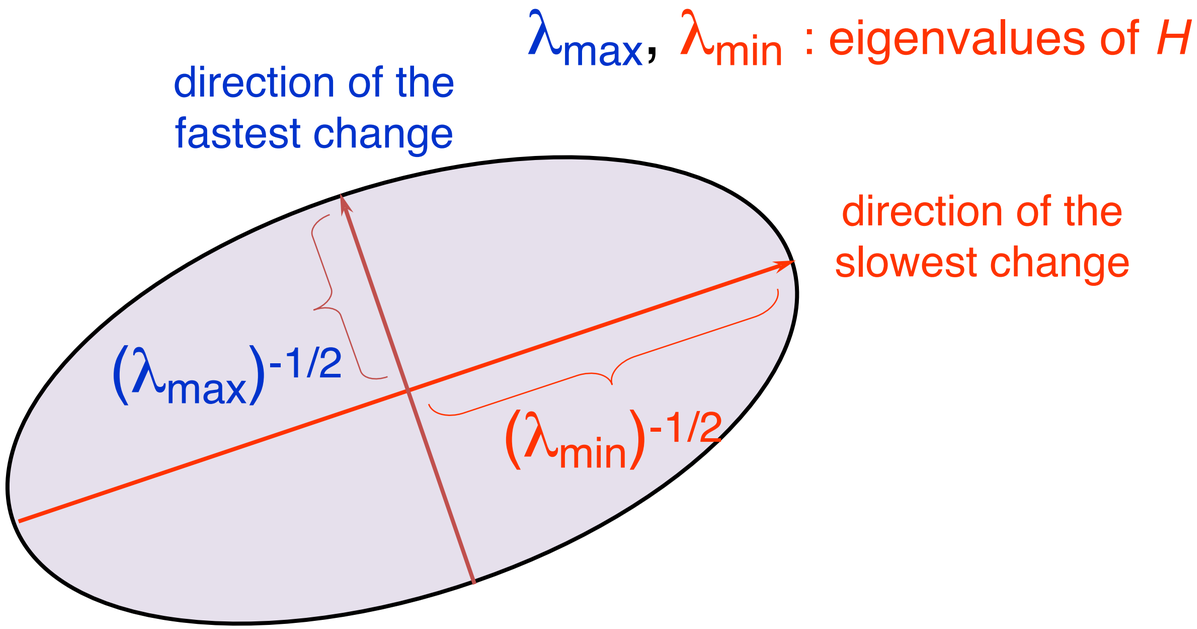


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

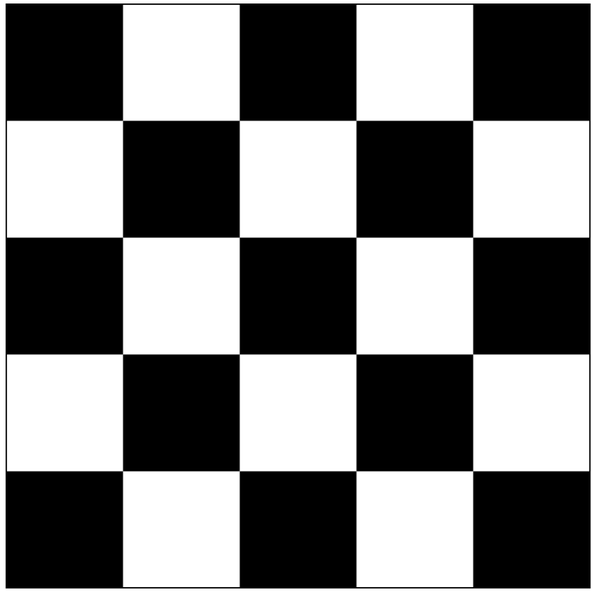
Corner detection: the math

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

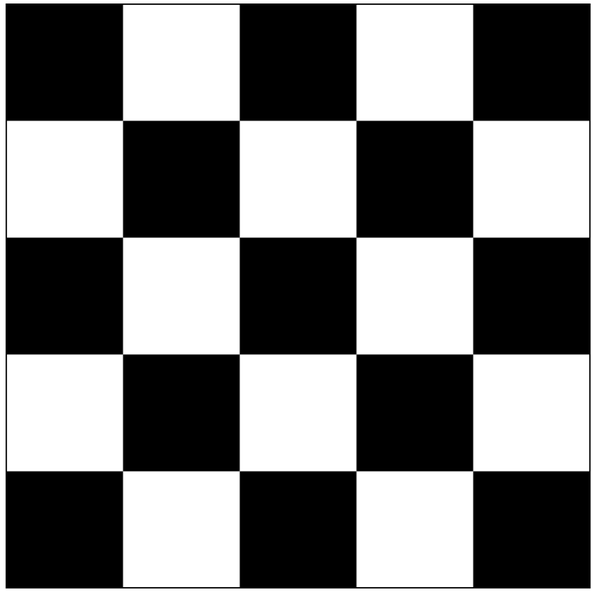
Corner detection: the math

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

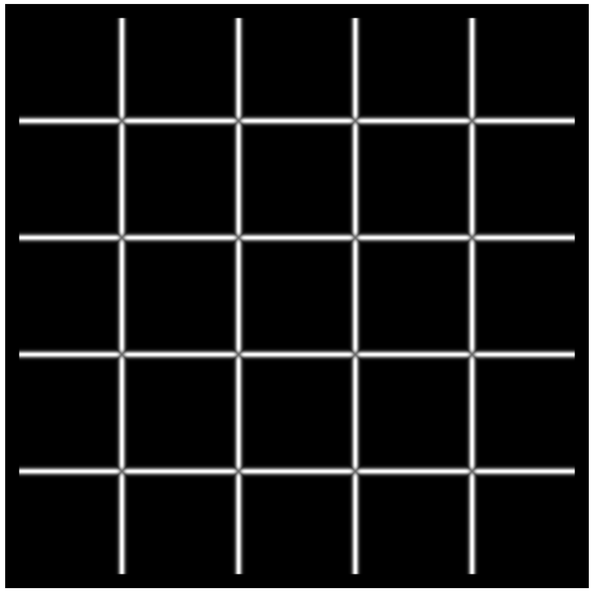
- What's our feature scoring function?

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

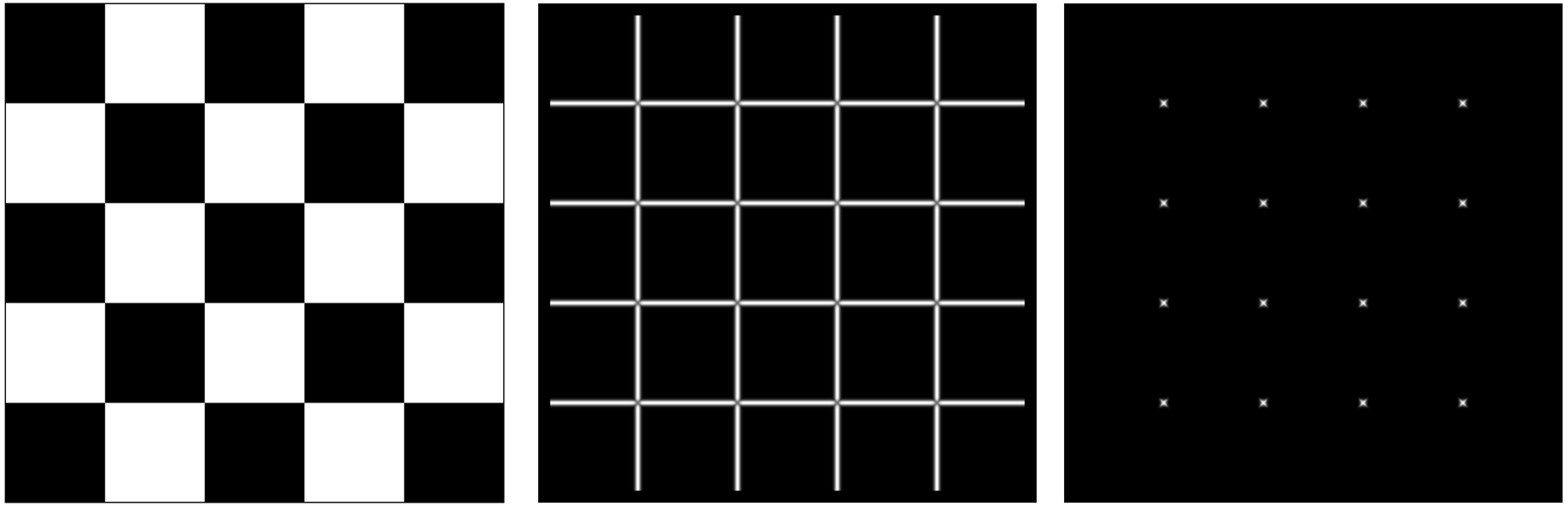
Corner detection: the math

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



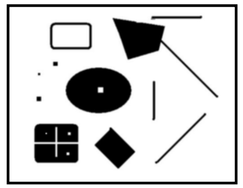
I λ_{max} λ_{min}

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

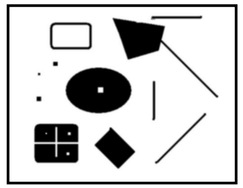


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

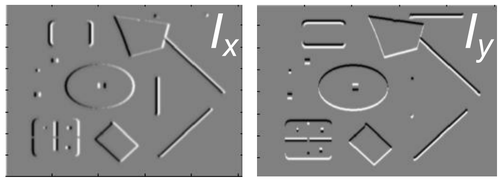
Harris Detector – Overview [Harris88]

Second moment matrix

0. Input image



1. Image derivatives

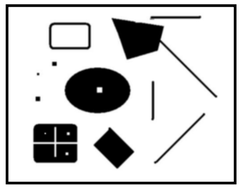


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

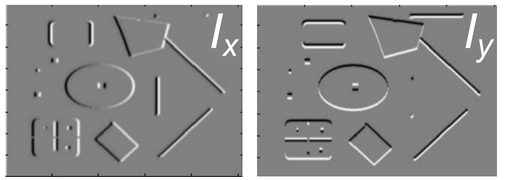
Harris Detector – Overview [Harris88]

Second moment matrix

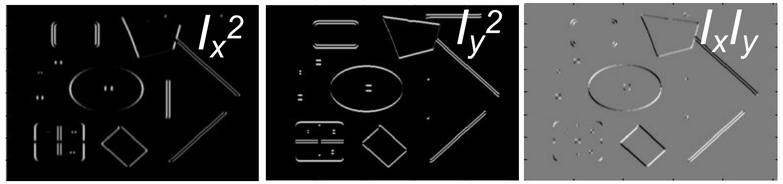
0. Input image



1. Image derivatives



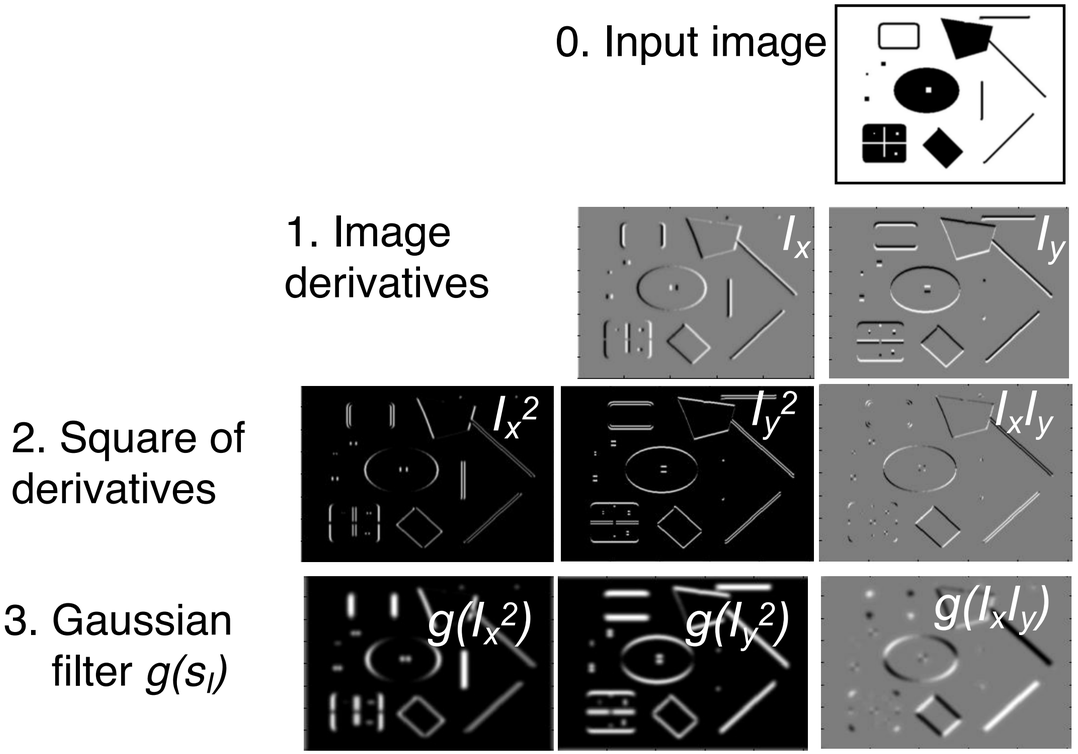
2. Square of derivatives



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

Harris Detector – Overview [Harris88]

Second moment matrix

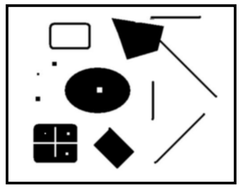


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

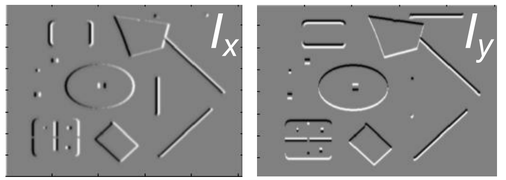
Harris Detector – Overview [Harris88]

Second moment matrix

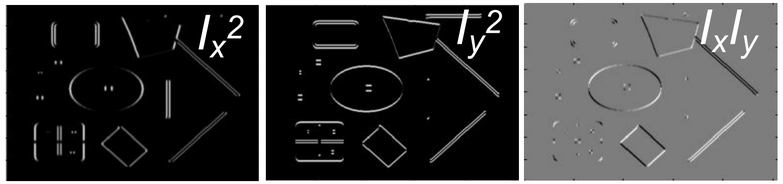
0. Input image



1. Image derivatives



2. Square of derivatives



3. Gaussian filter $g(s_I)$

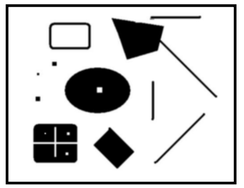
4. Cornerness function – both eigenvalues are strong

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

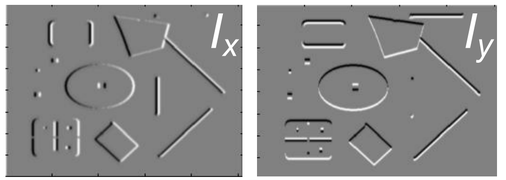
Harris Detector – Overview [Harris88]

Second moment matrix

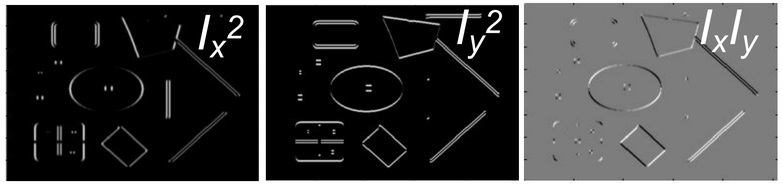
0. Input image



1. Image derivatives



2. Square of derivatives



3. Gaussian filter $g(s_I)$

4. Cornerness function – both eigenvalues are strong

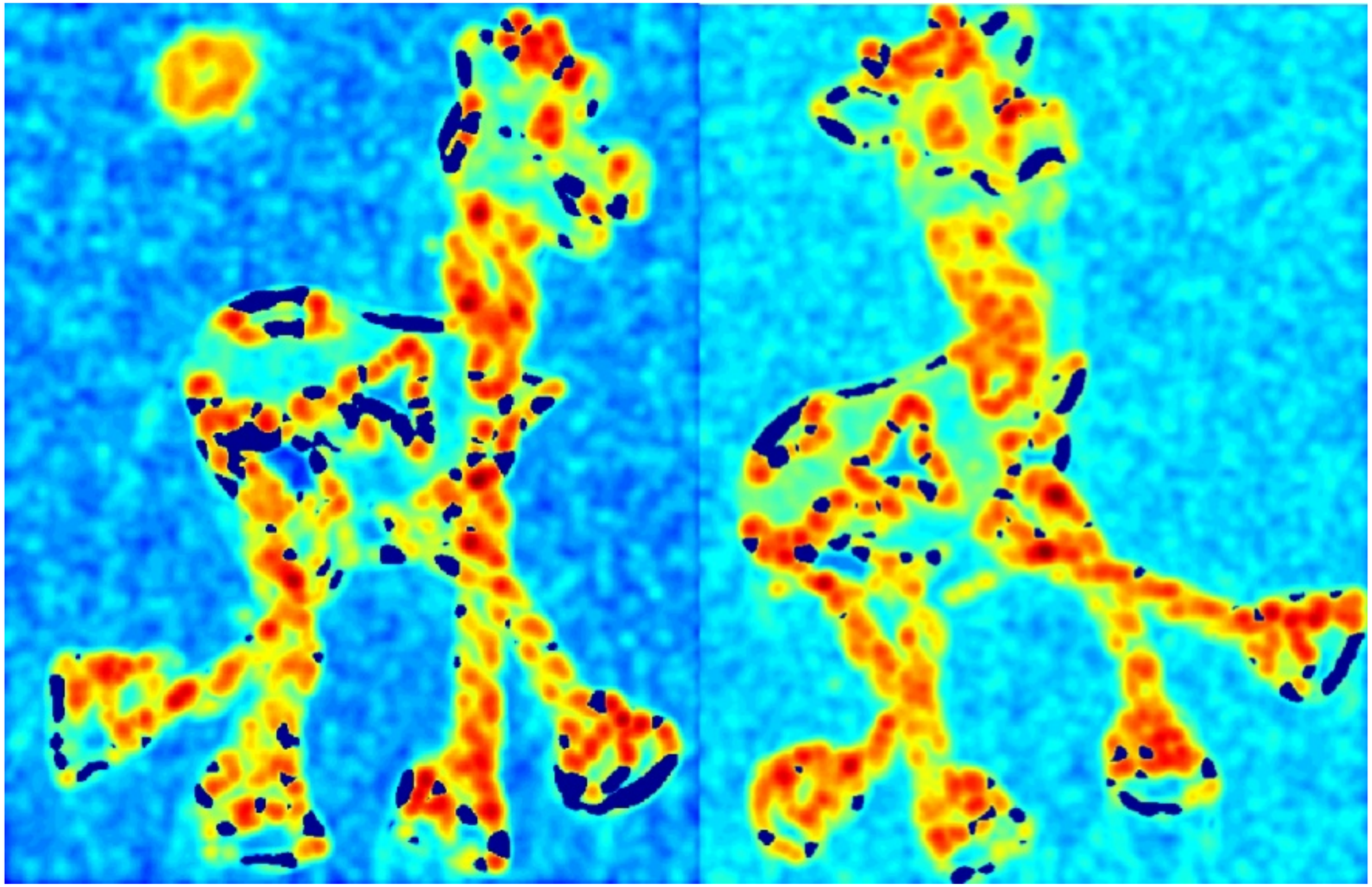
5. Non-maxima suppression

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

Harris detector example

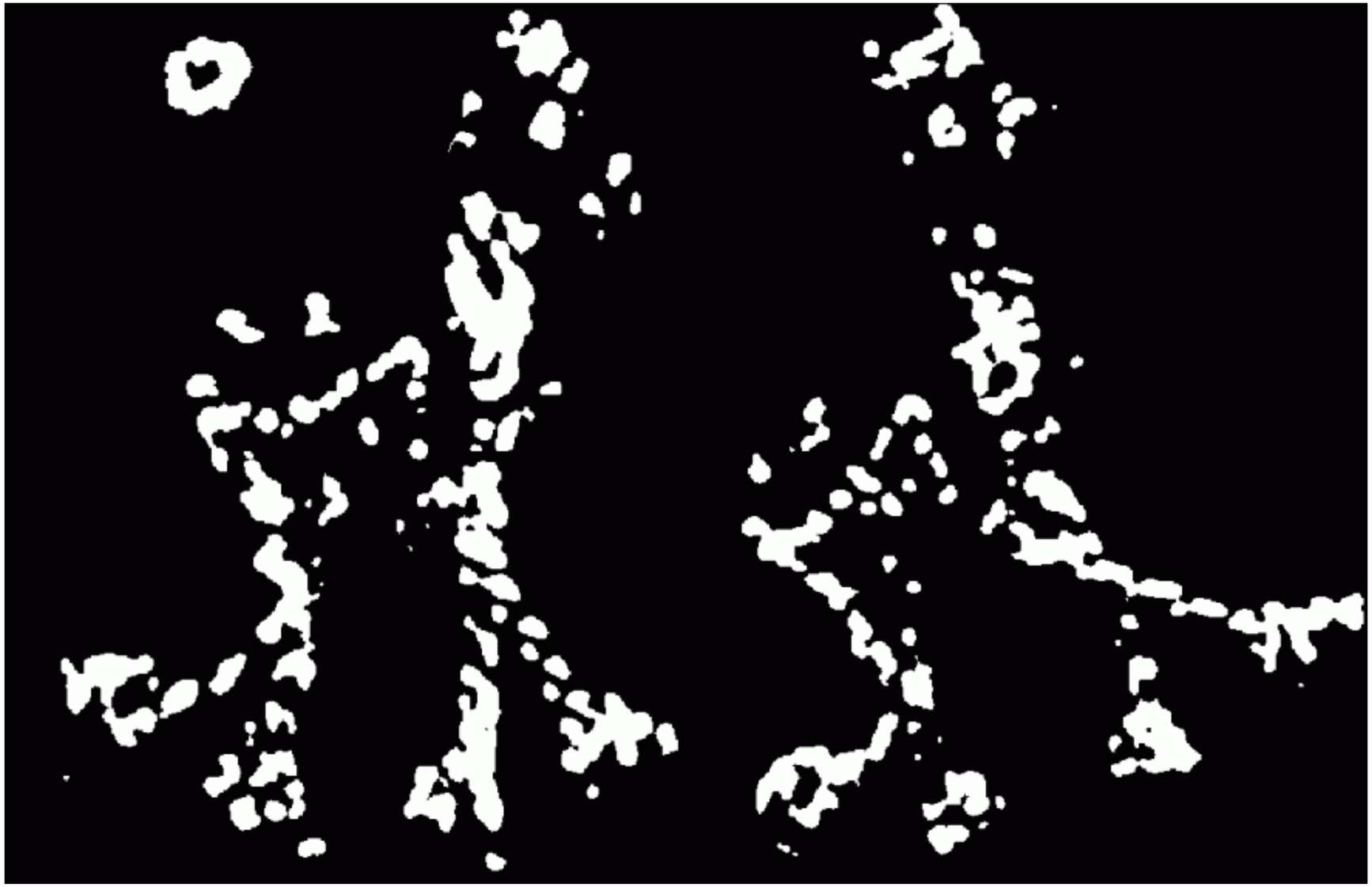
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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)

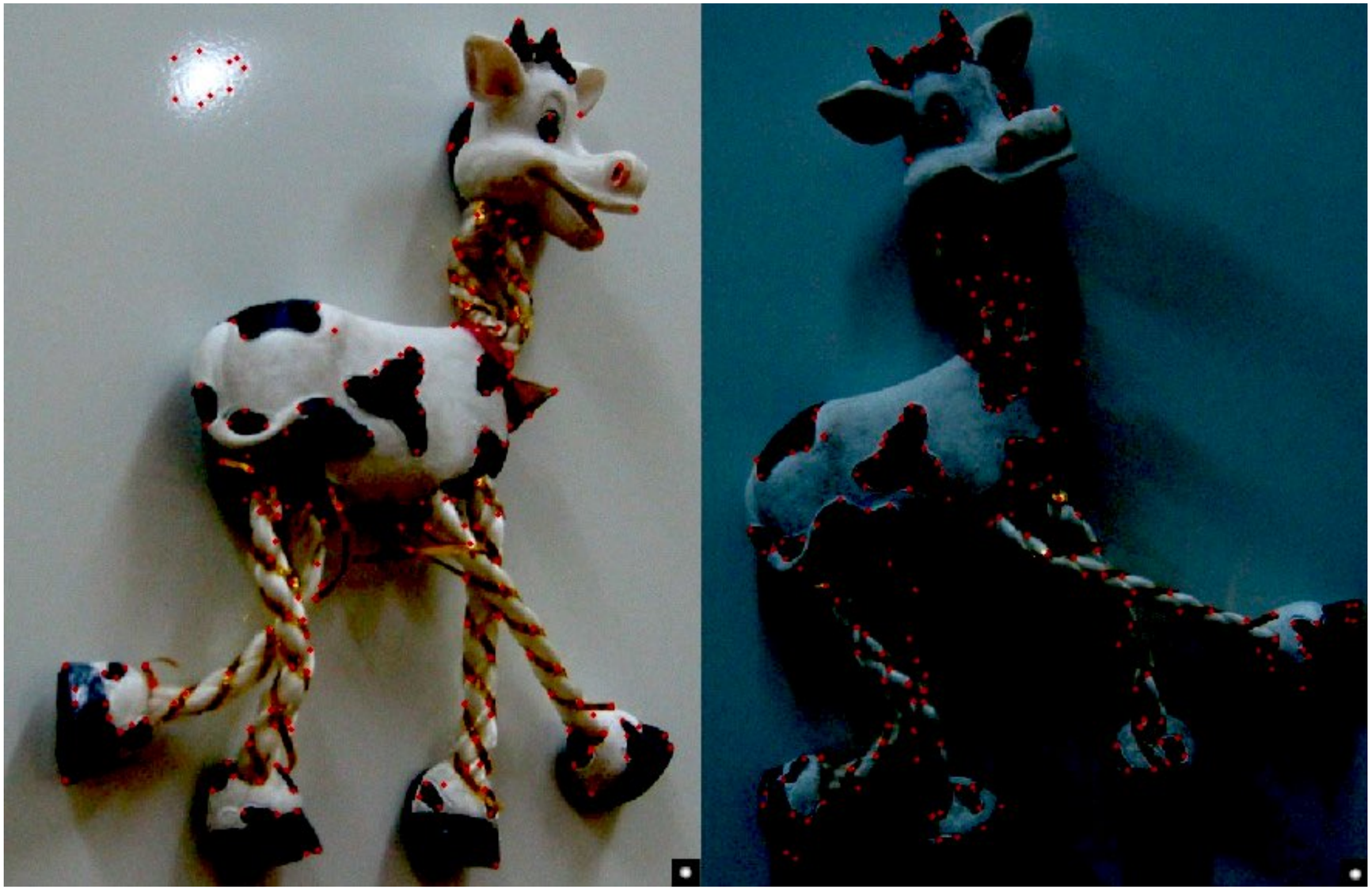


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Find local maxima of f (non-max suppression)

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

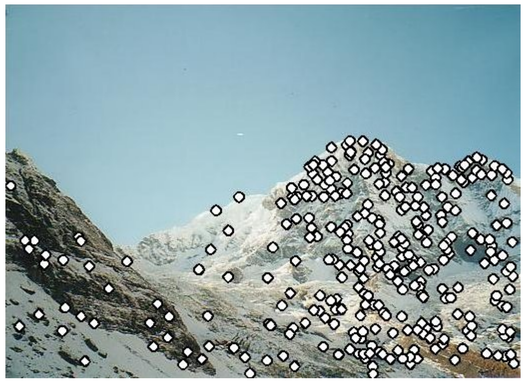
Harris features (in red)



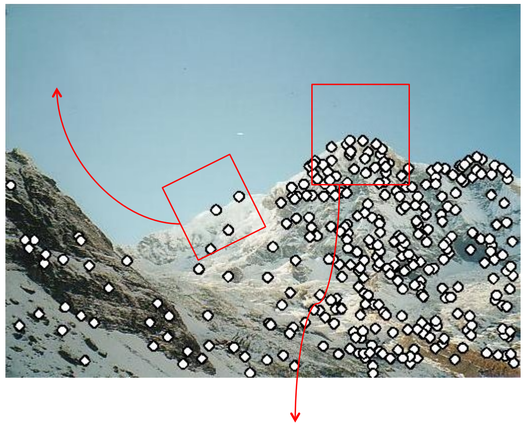
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Local features: main components

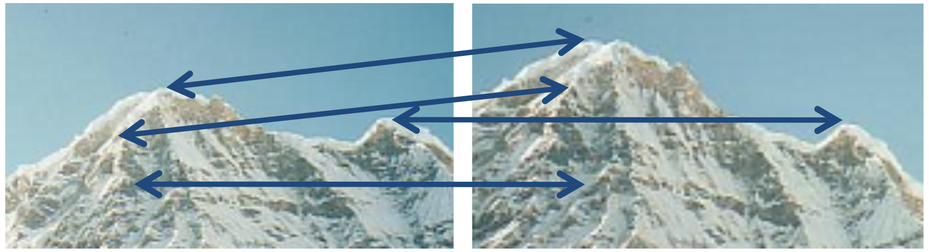
1) Detection: Identify the interest points



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



3) Matching: Determine correspondence between descriptors in two views

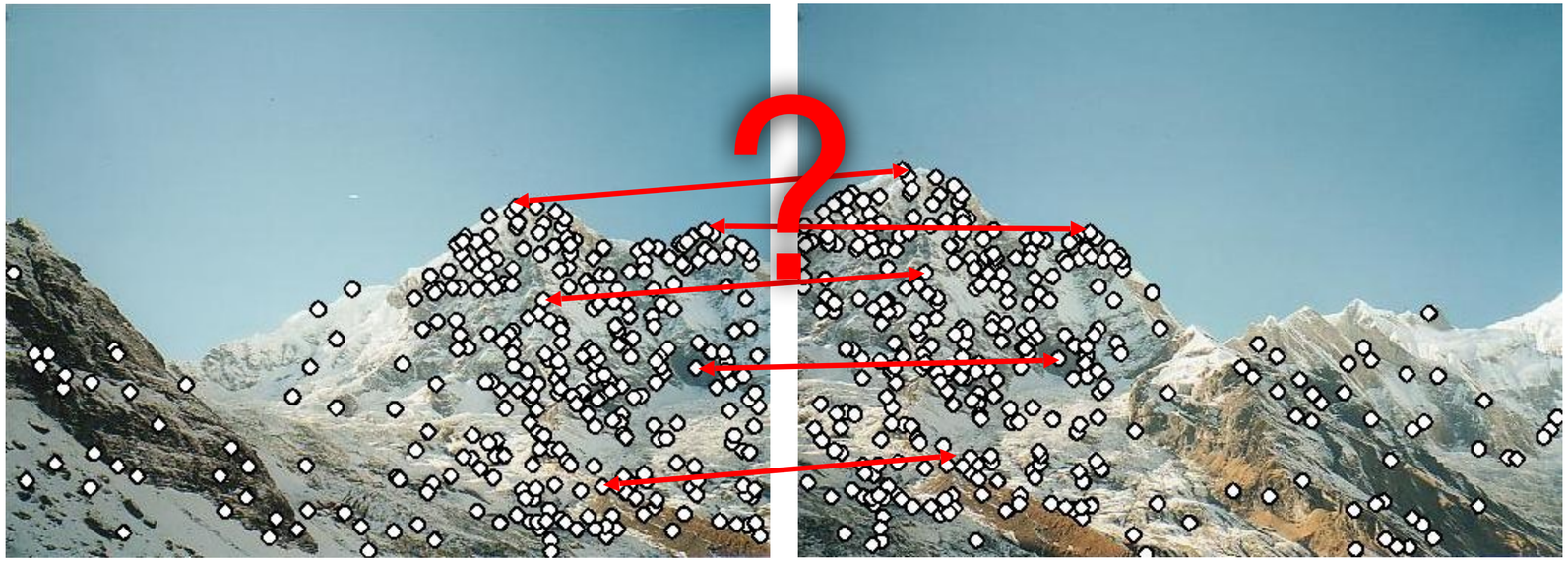


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Kristen Grauman

Feature descriptors

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

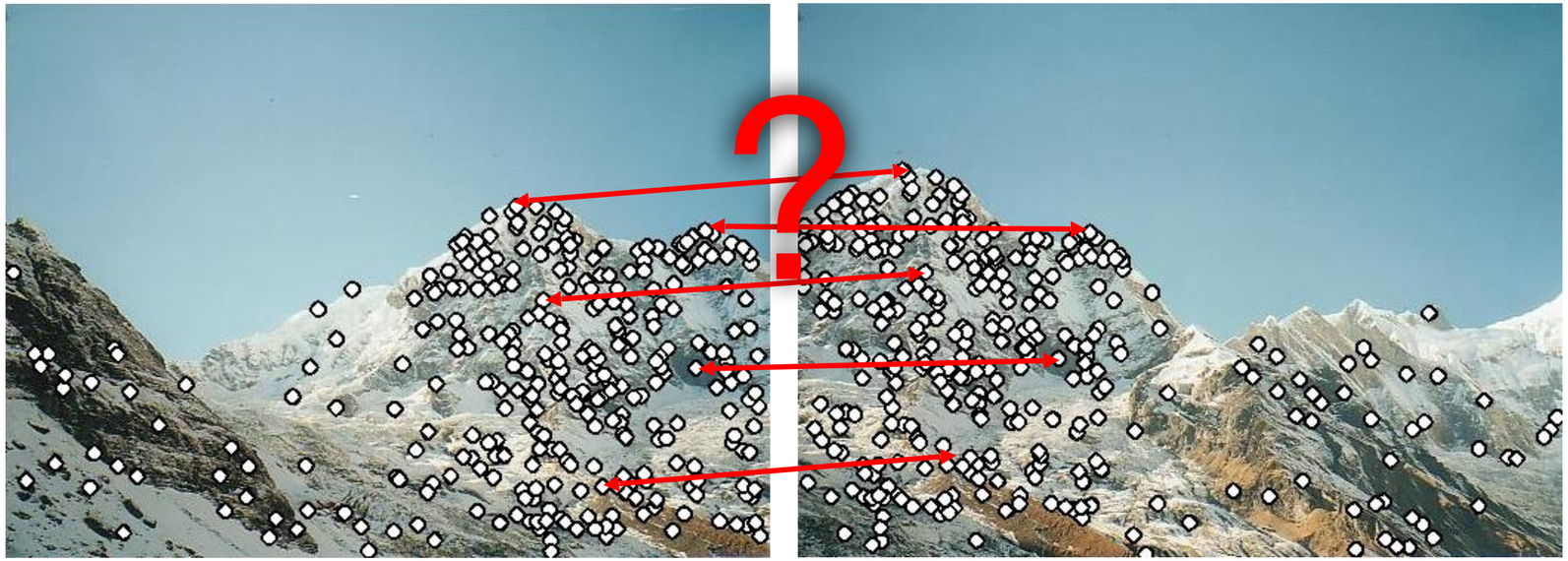


Answer: Come up with a *descriptor* for each point, find similar descriptors between the two images

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

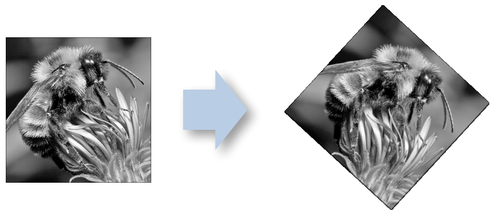
- Discriminability:
  - Descriptor should be highly unique for each point

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

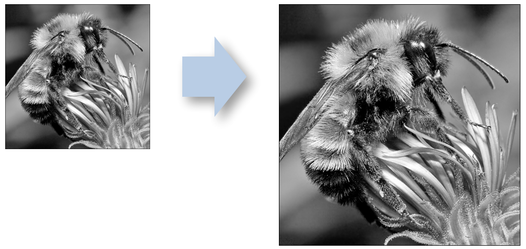
Image transformations revisited

- Geometric

Rotation

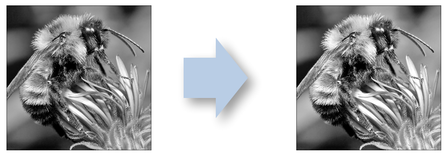


Scale



- 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

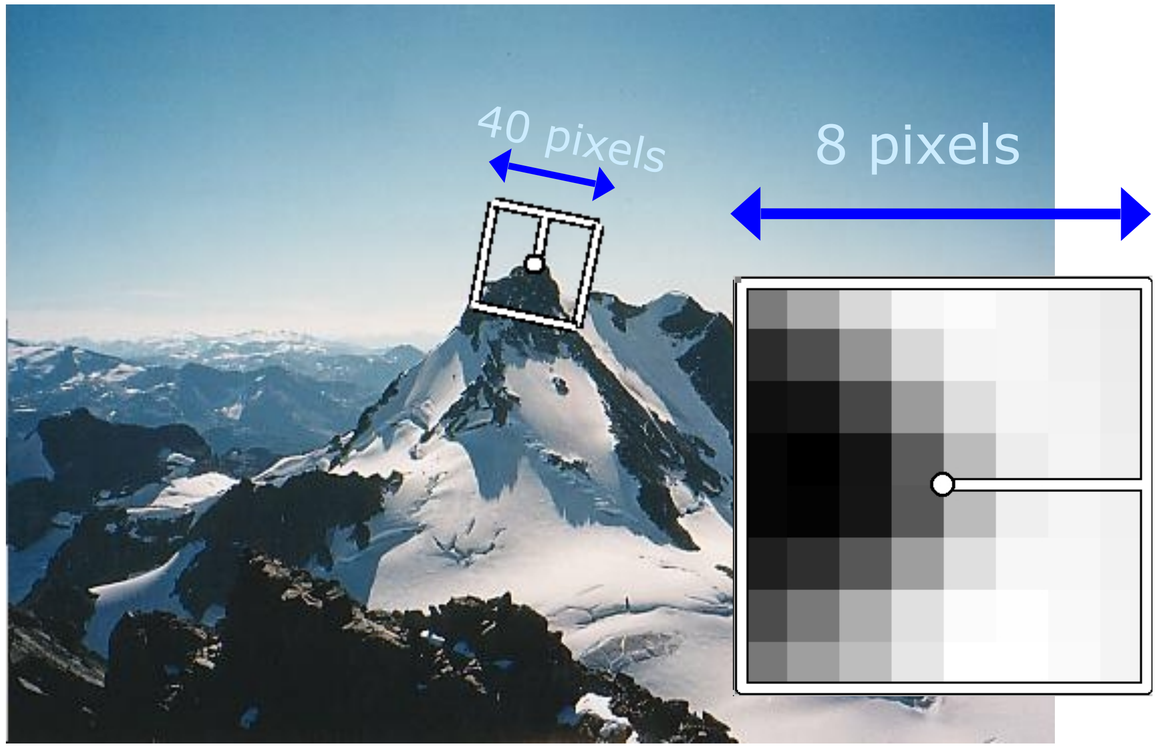
Figure by Matthew Brown

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

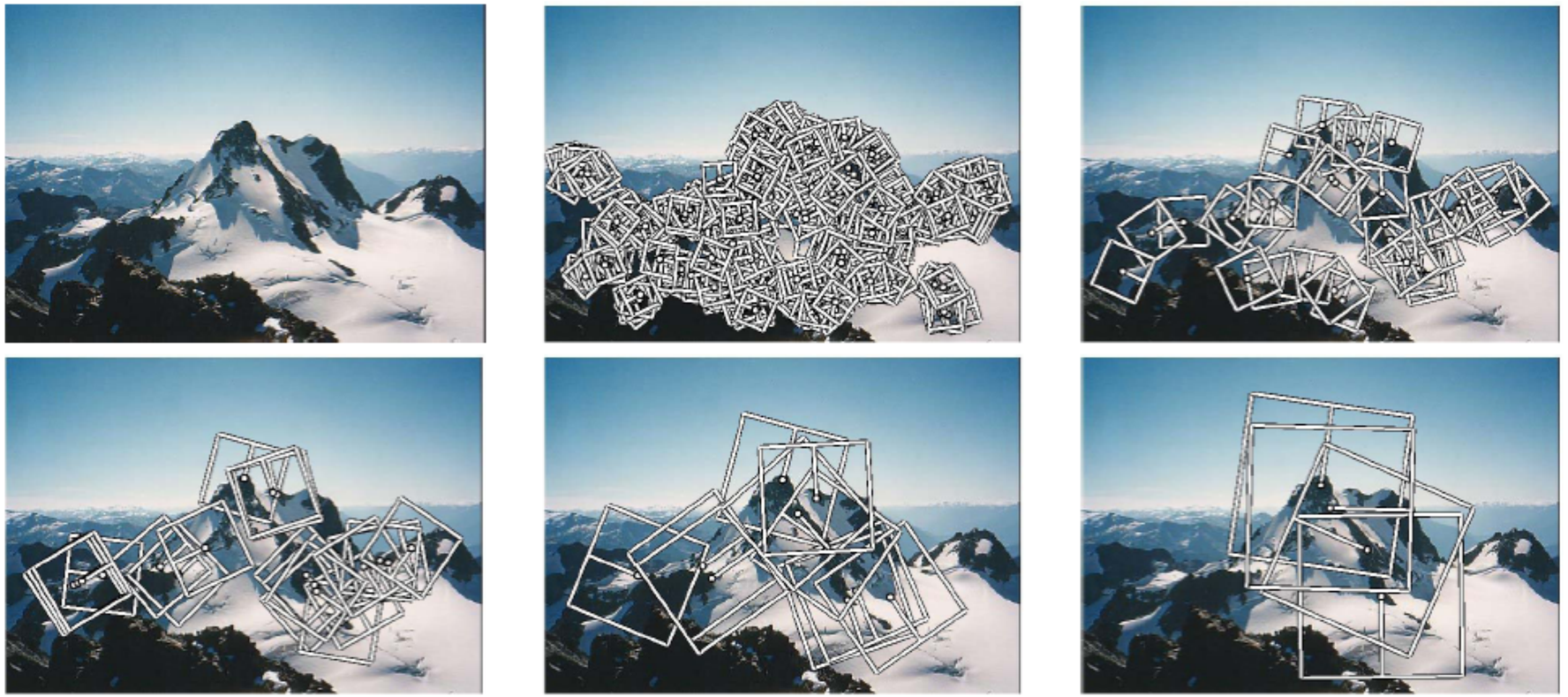


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

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

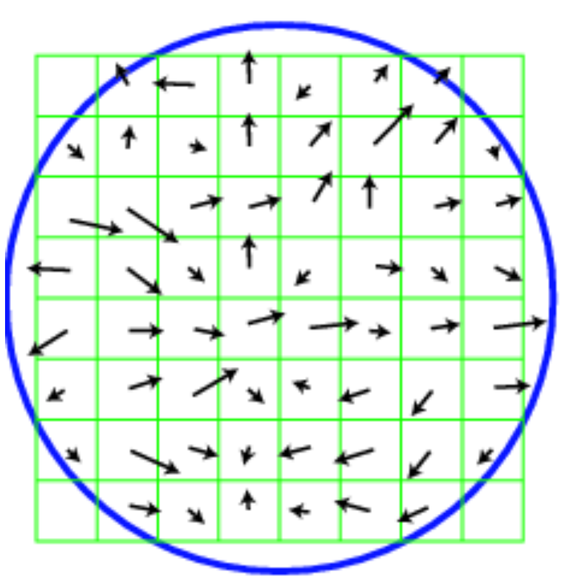
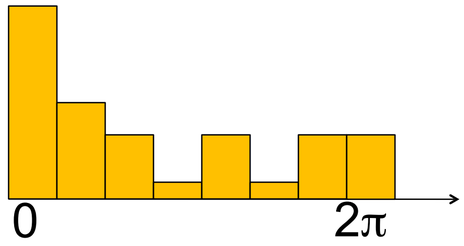
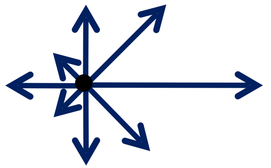


Image gradients



angle histogram



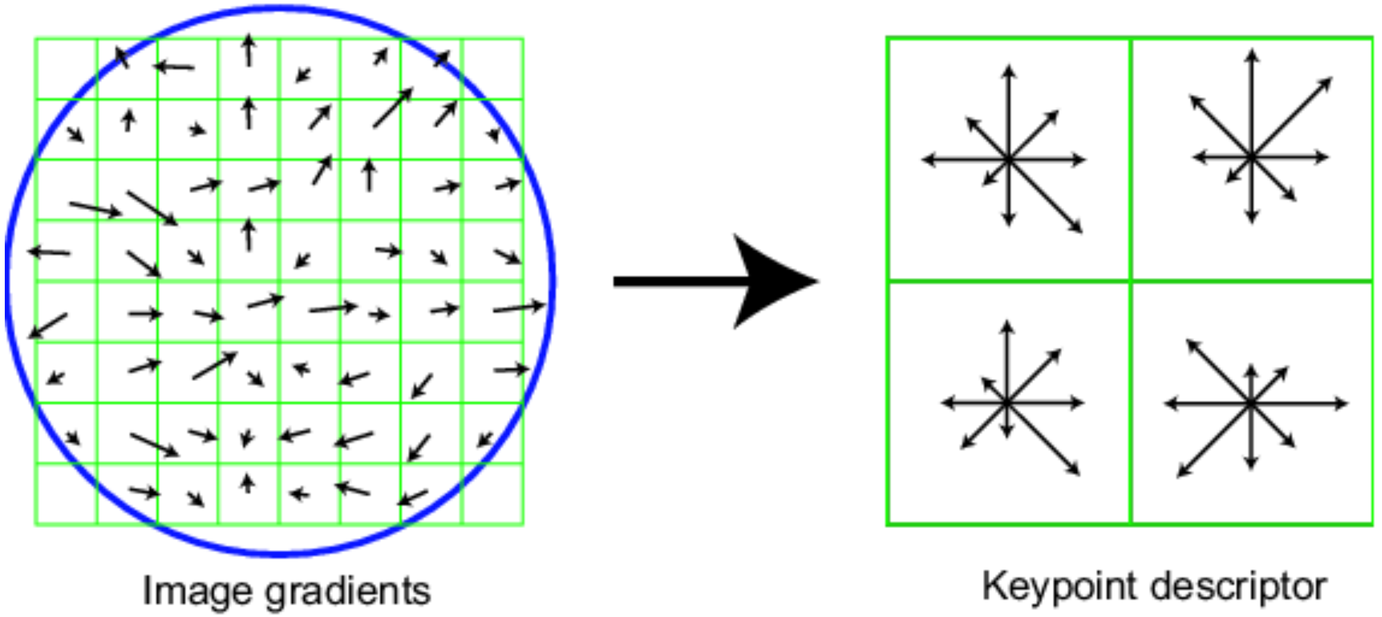
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

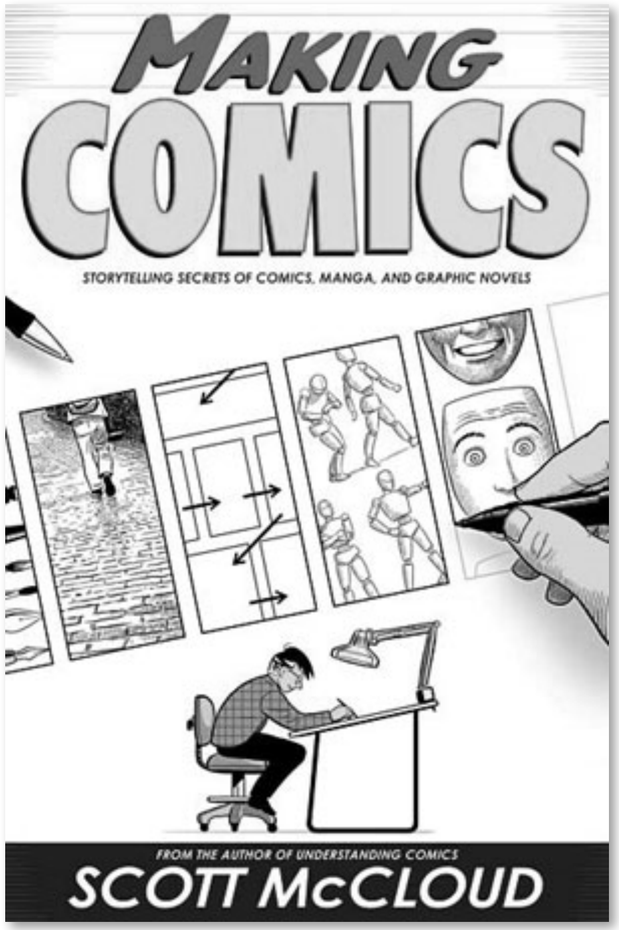
Properties of SIFT

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

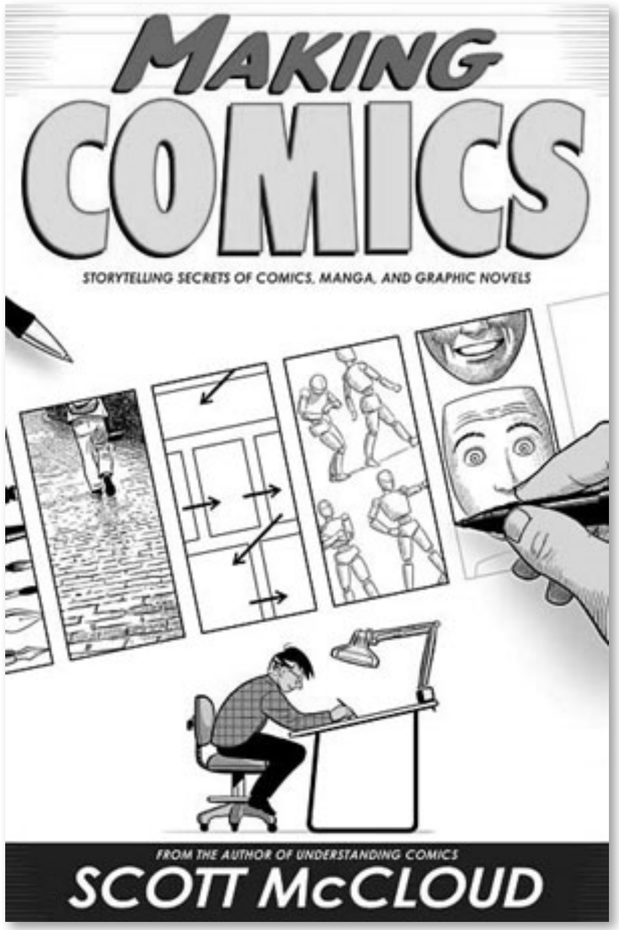
Slide credit: Noah Snavely, CS 670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

SIFT Example

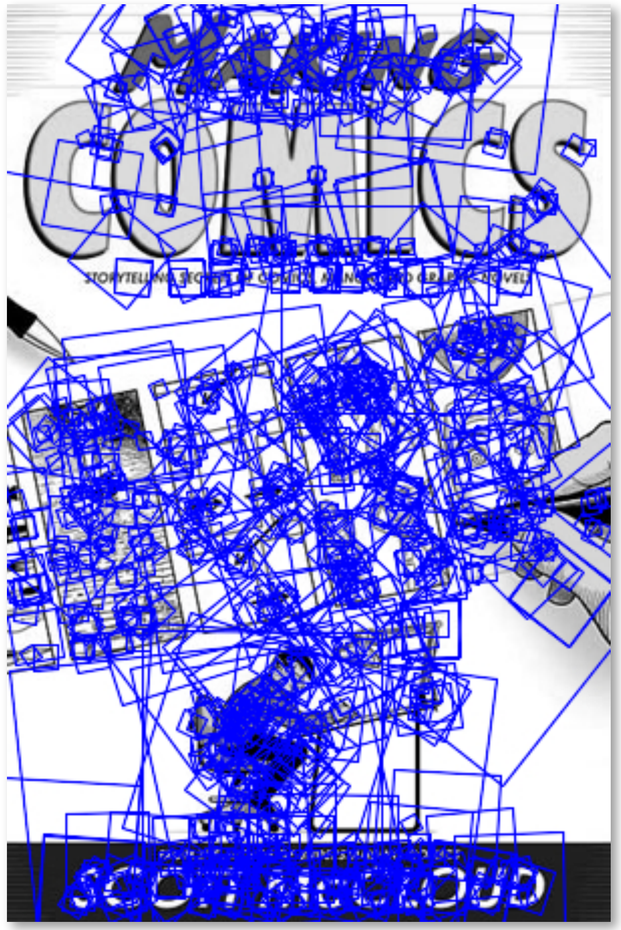


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

SIFT Example



sift

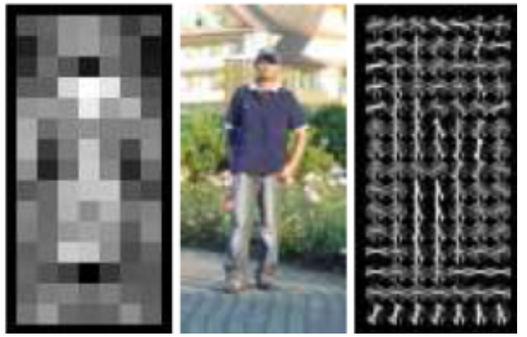


868 SIFT features

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

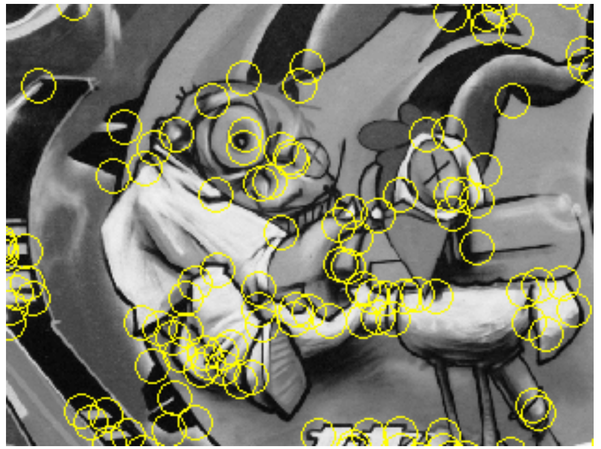
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Questions?

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

Summary

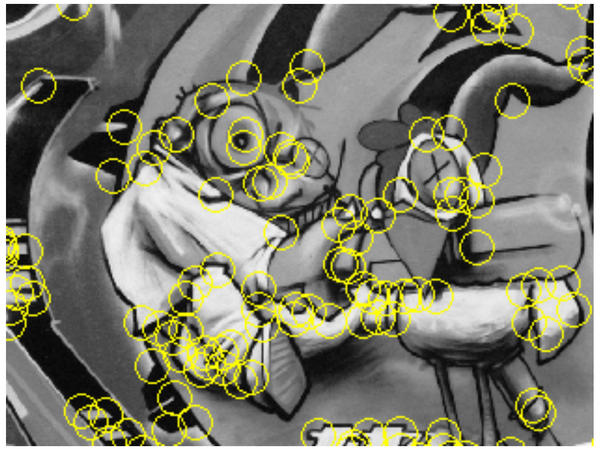
- Keypoint detection: repeatable and distinctive
  - Corners, blobs
  - Harris, DoG



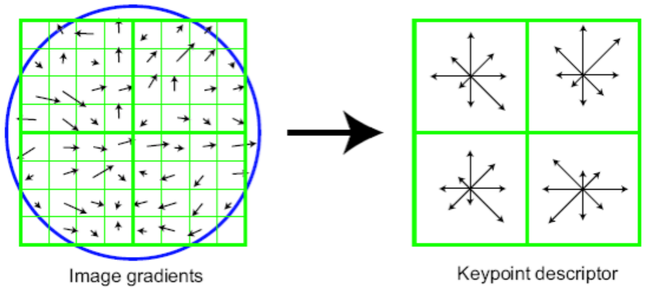
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Summary

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

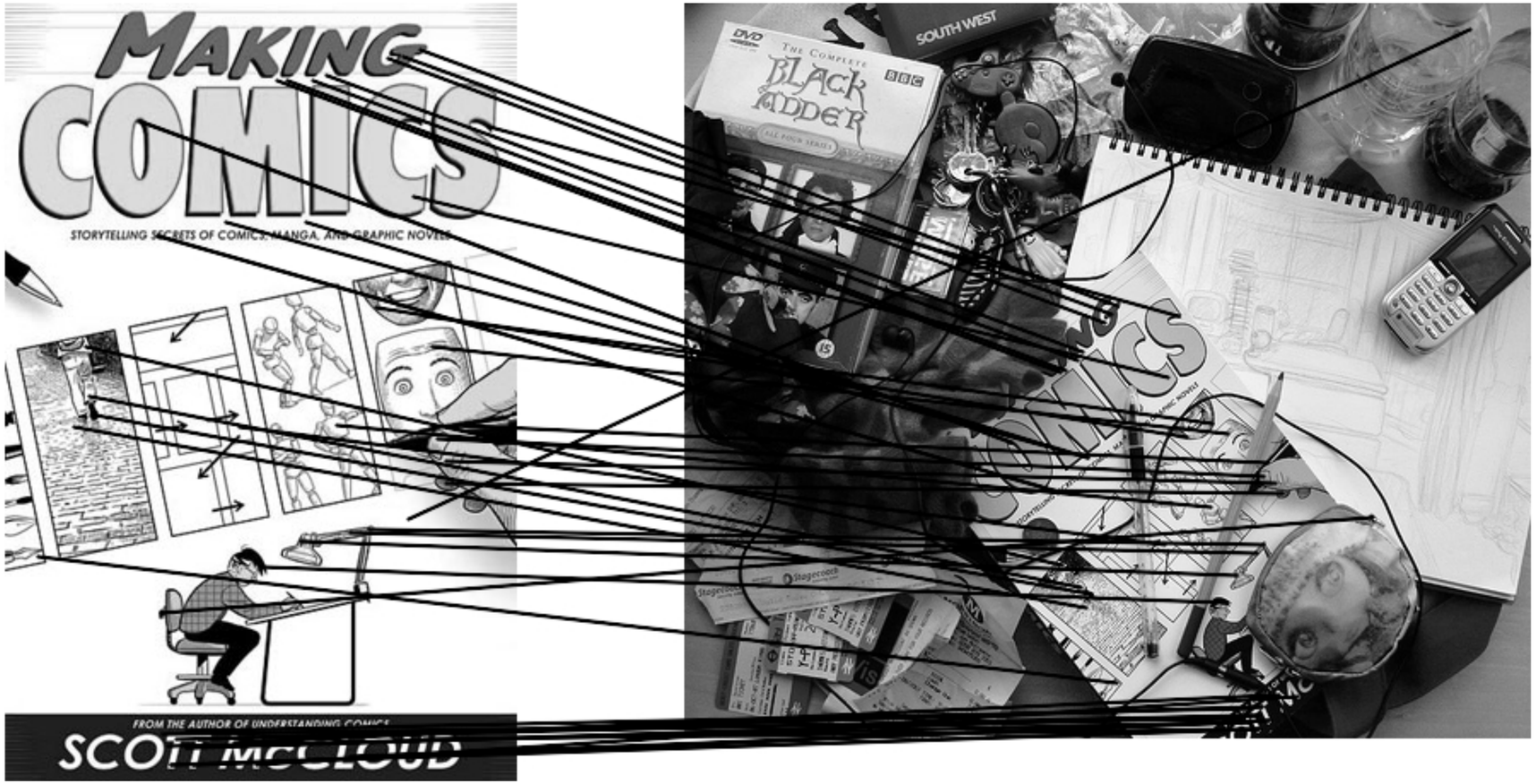


- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT and variants are typically good for stitching and recognition
  - But, need not stick to one



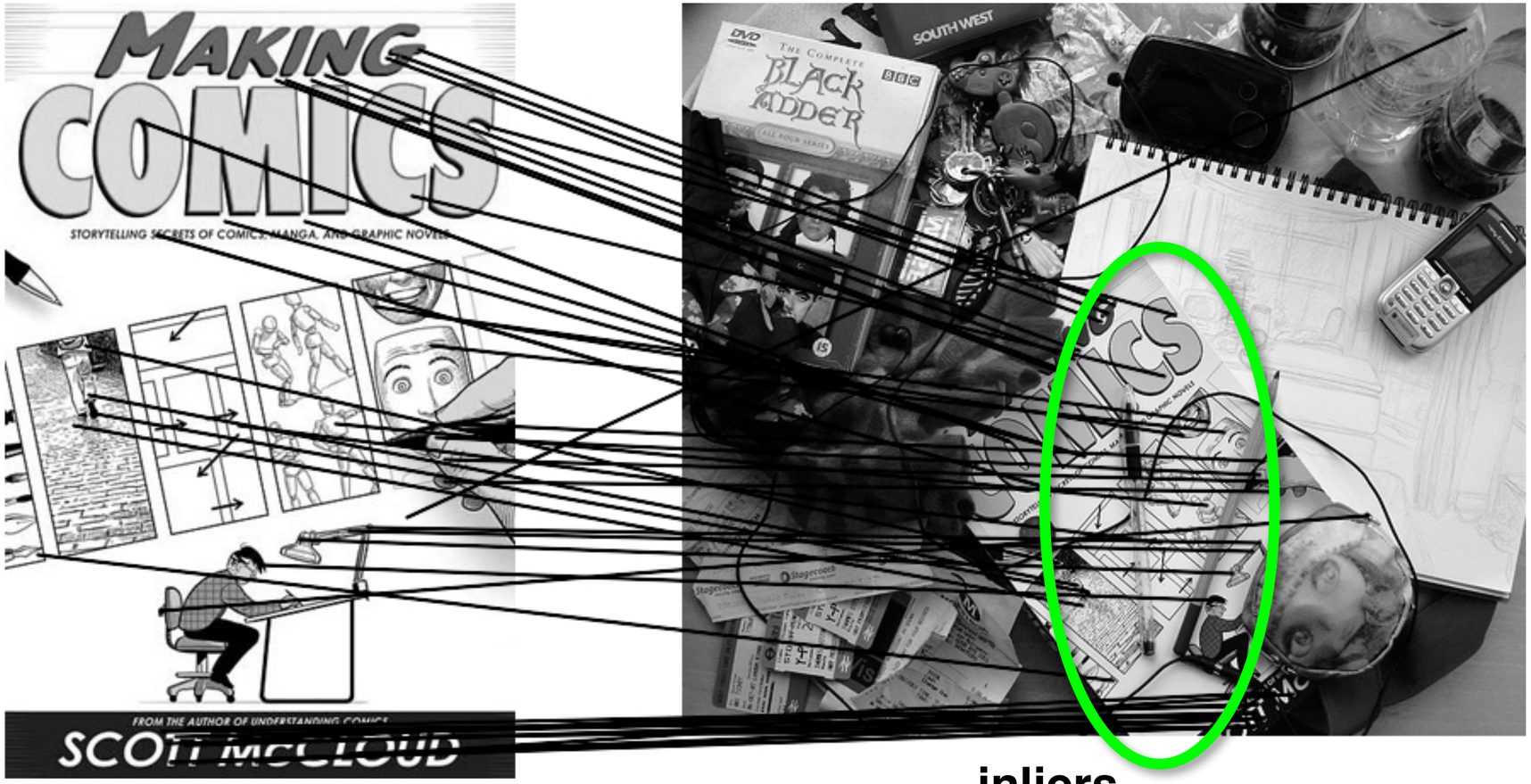
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Outliers



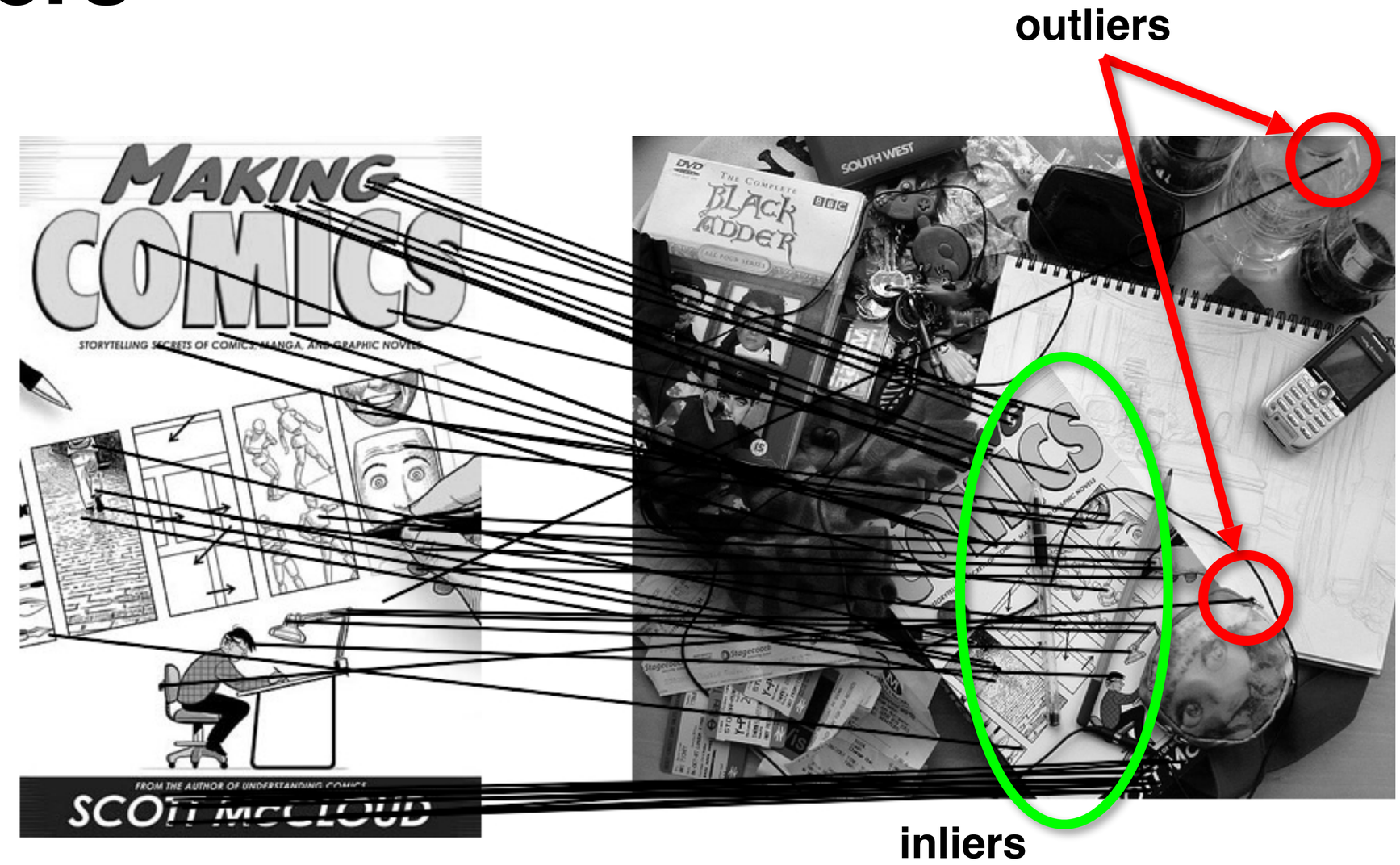
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Outliers



inliers

Outliers



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

Robustness

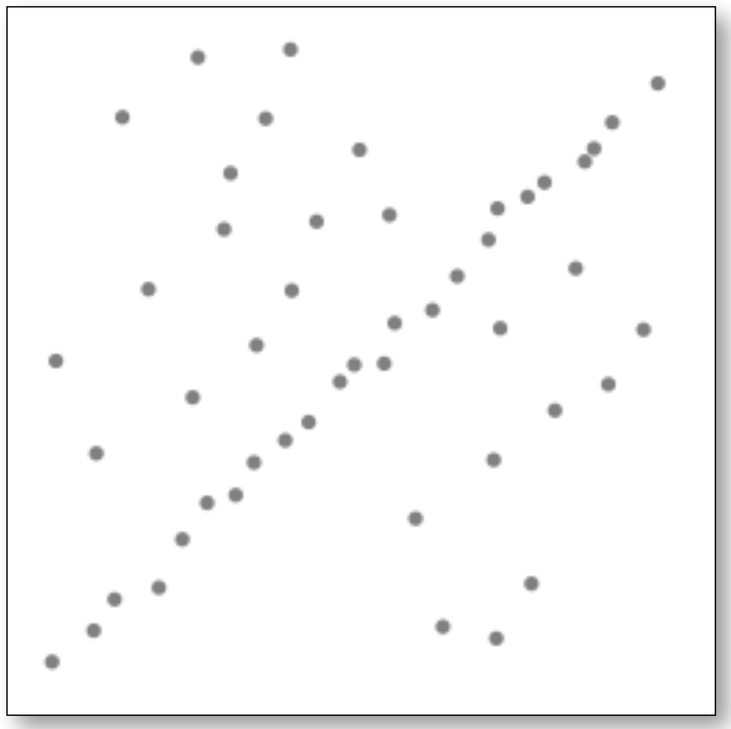
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Robustness

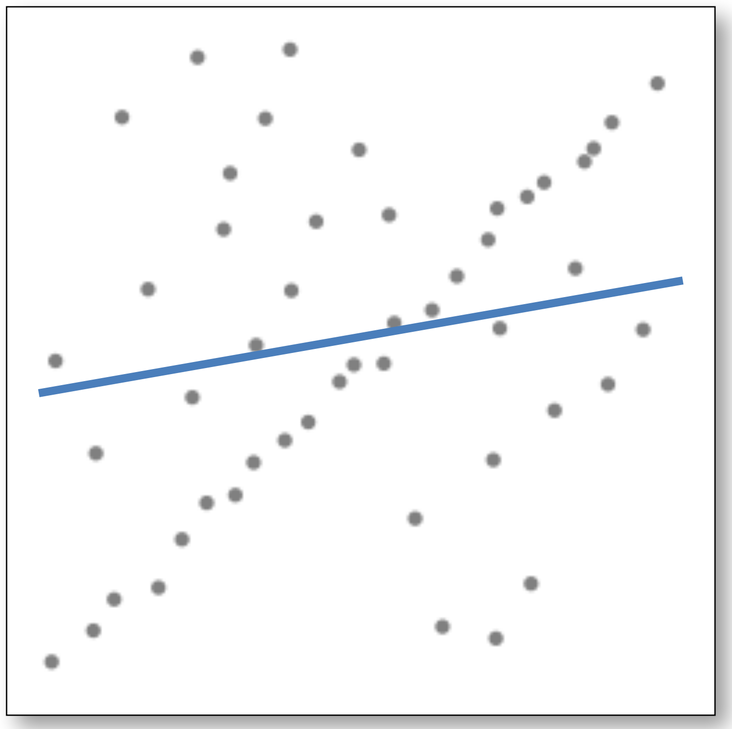
Problem: Fit a line to these datapoints

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

Robustness



Problem: Fit a line to these datapoints

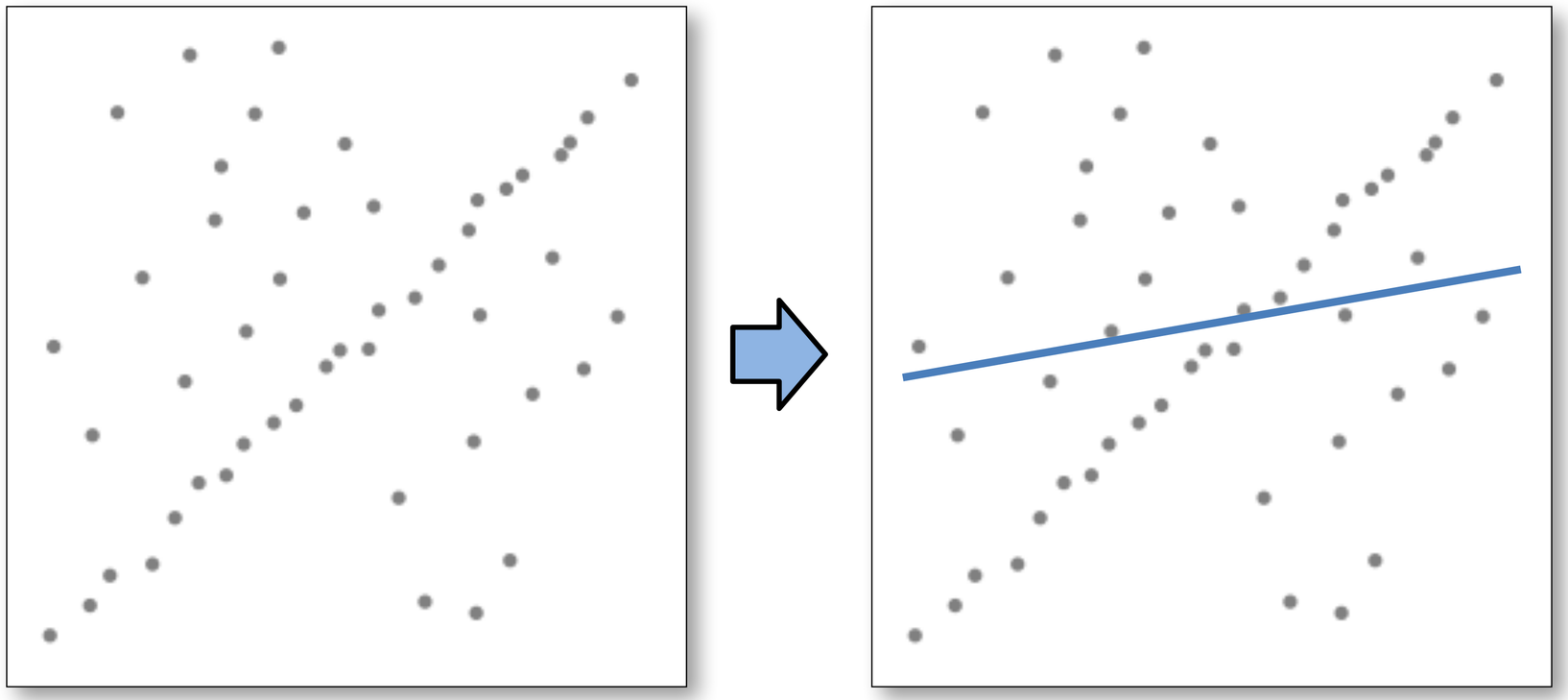


Least squares fit

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

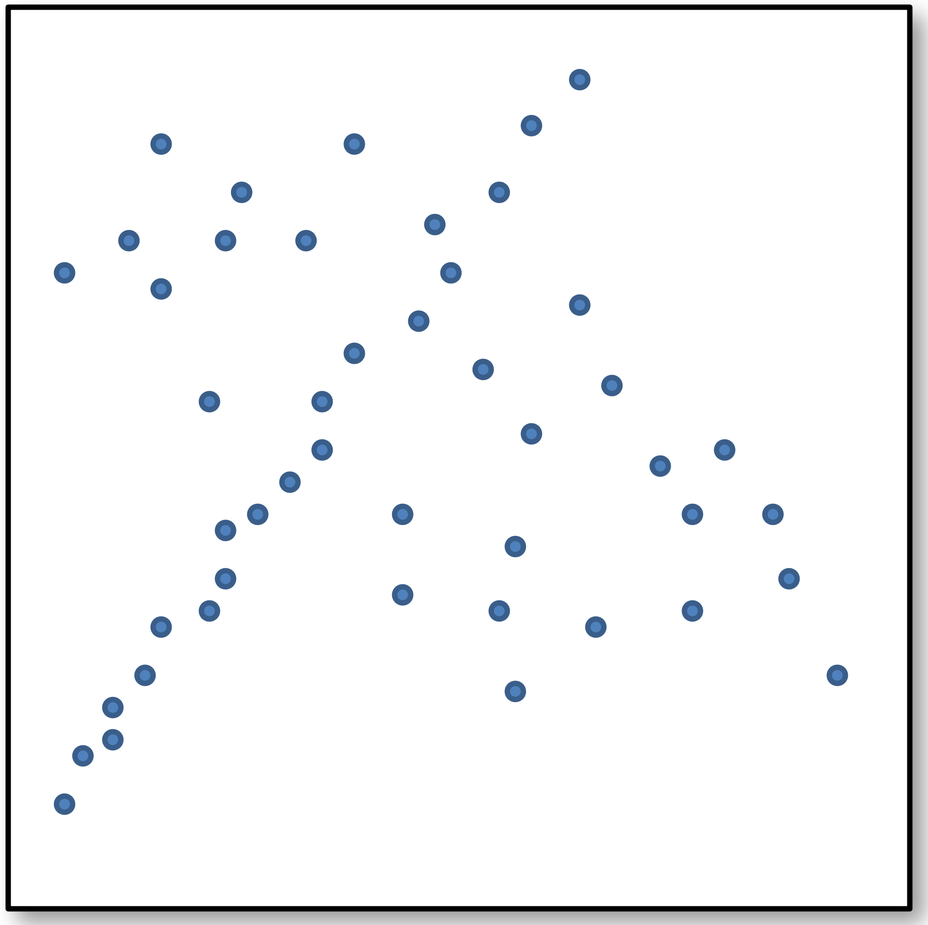
Idea

- 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

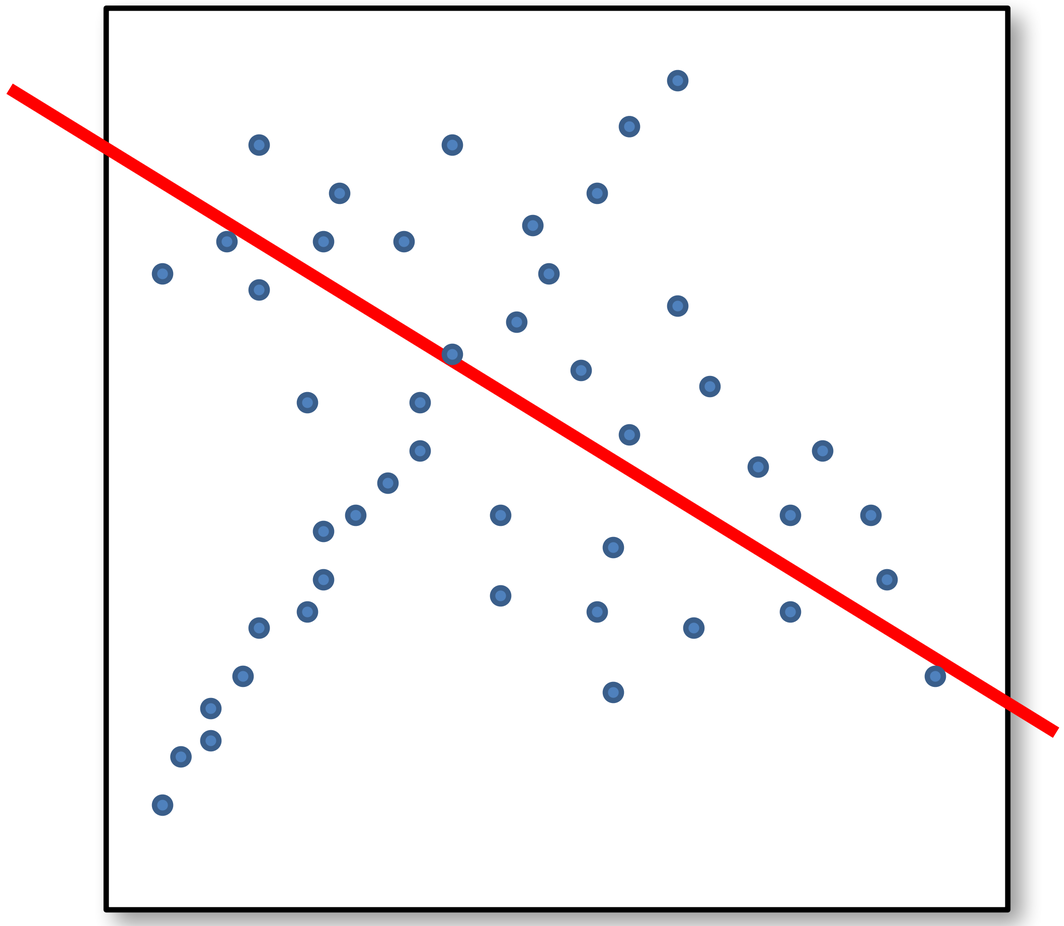
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



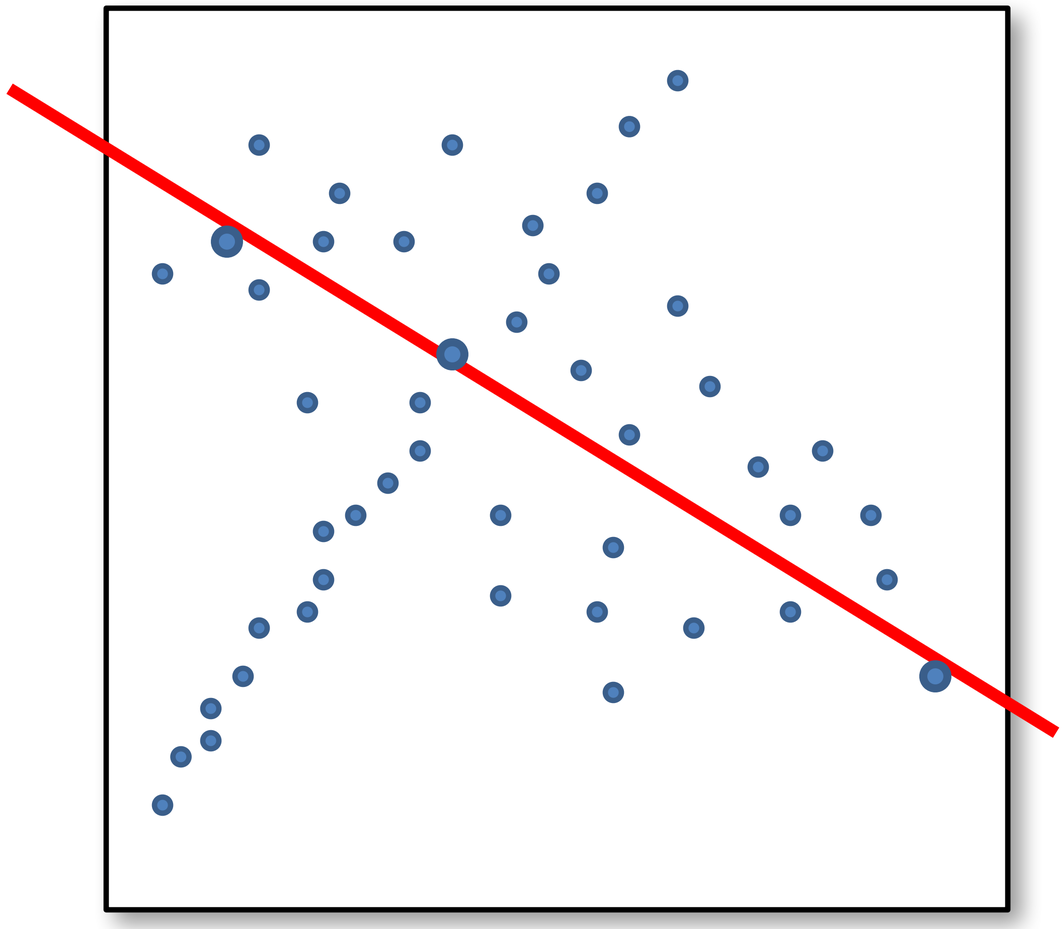
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



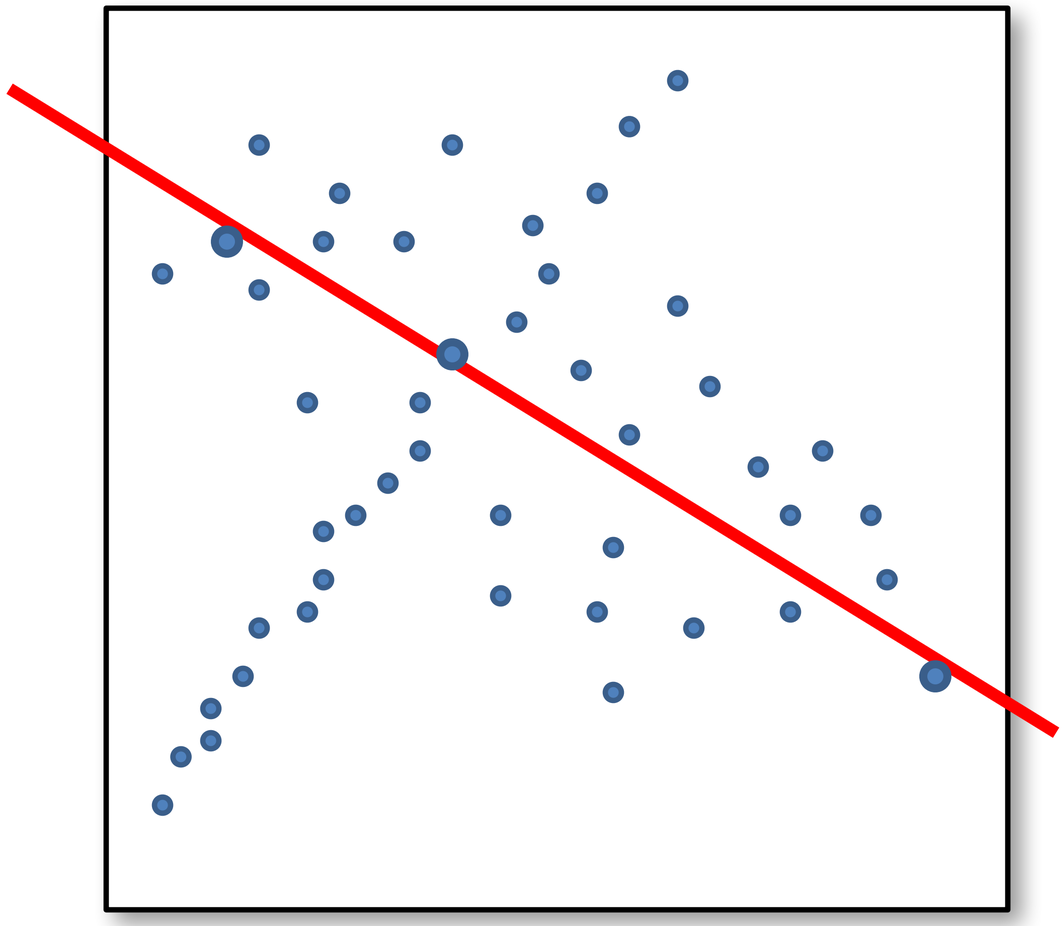
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



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

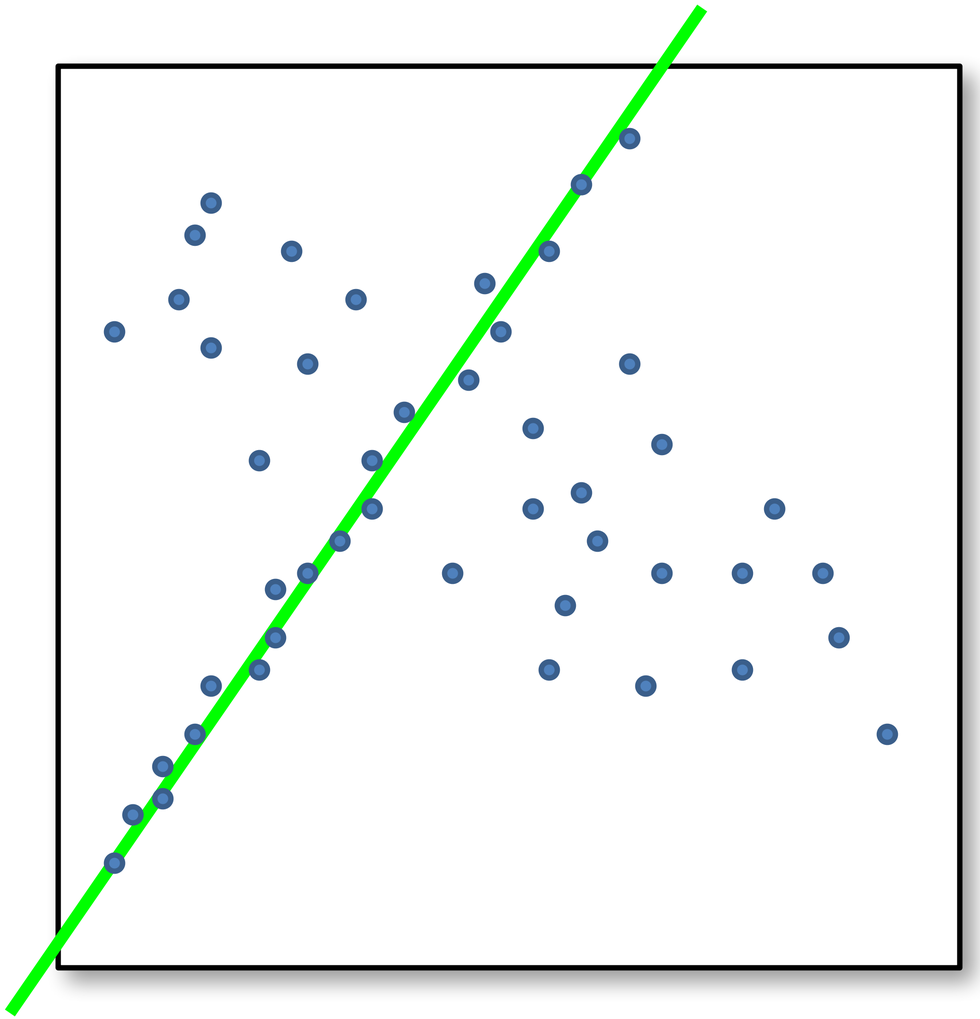
Counting inliers



Inliers: 3

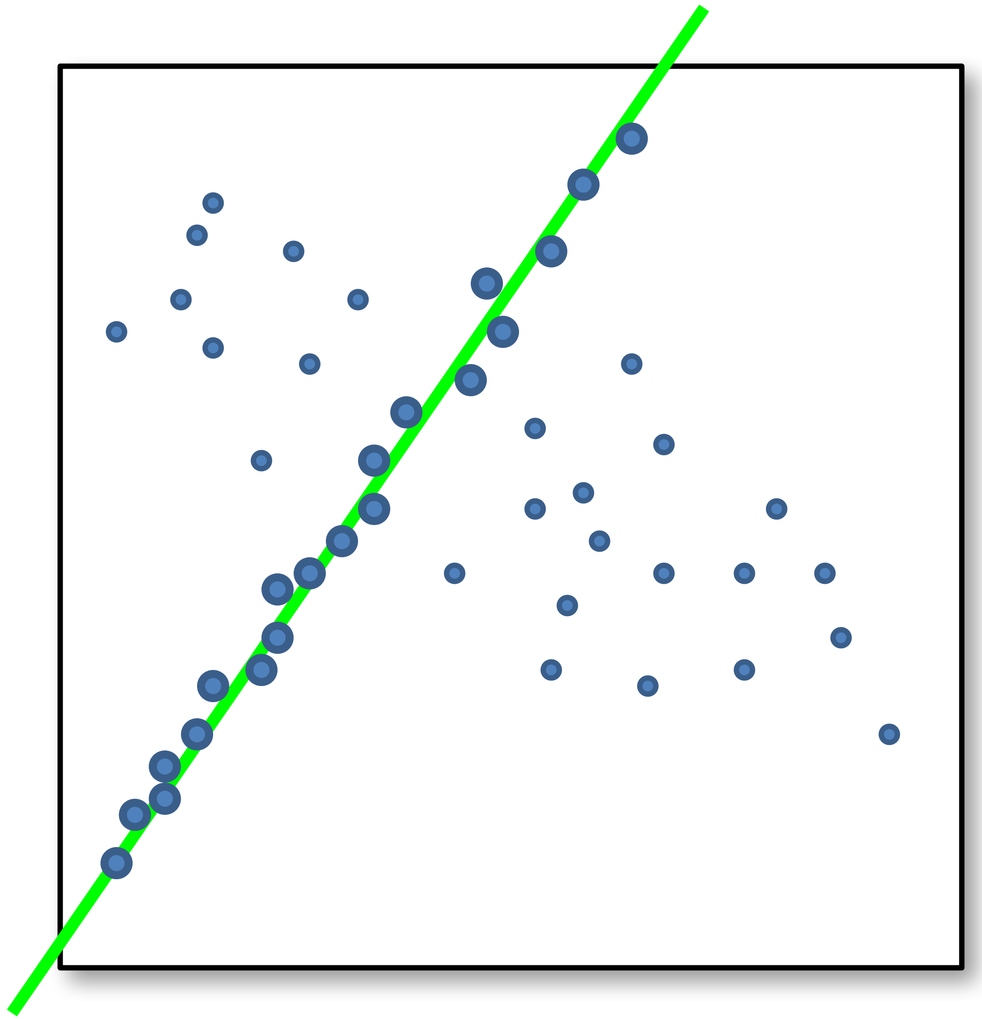
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Counting inliers



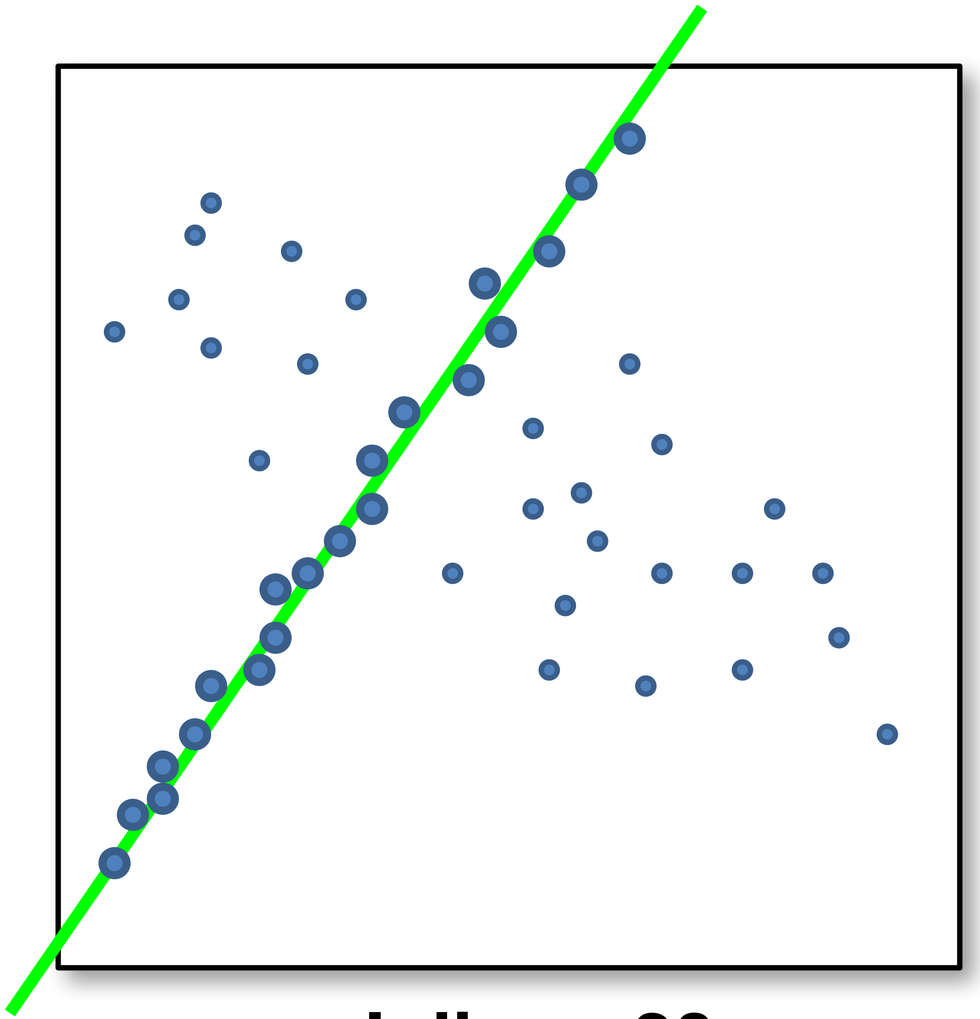
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

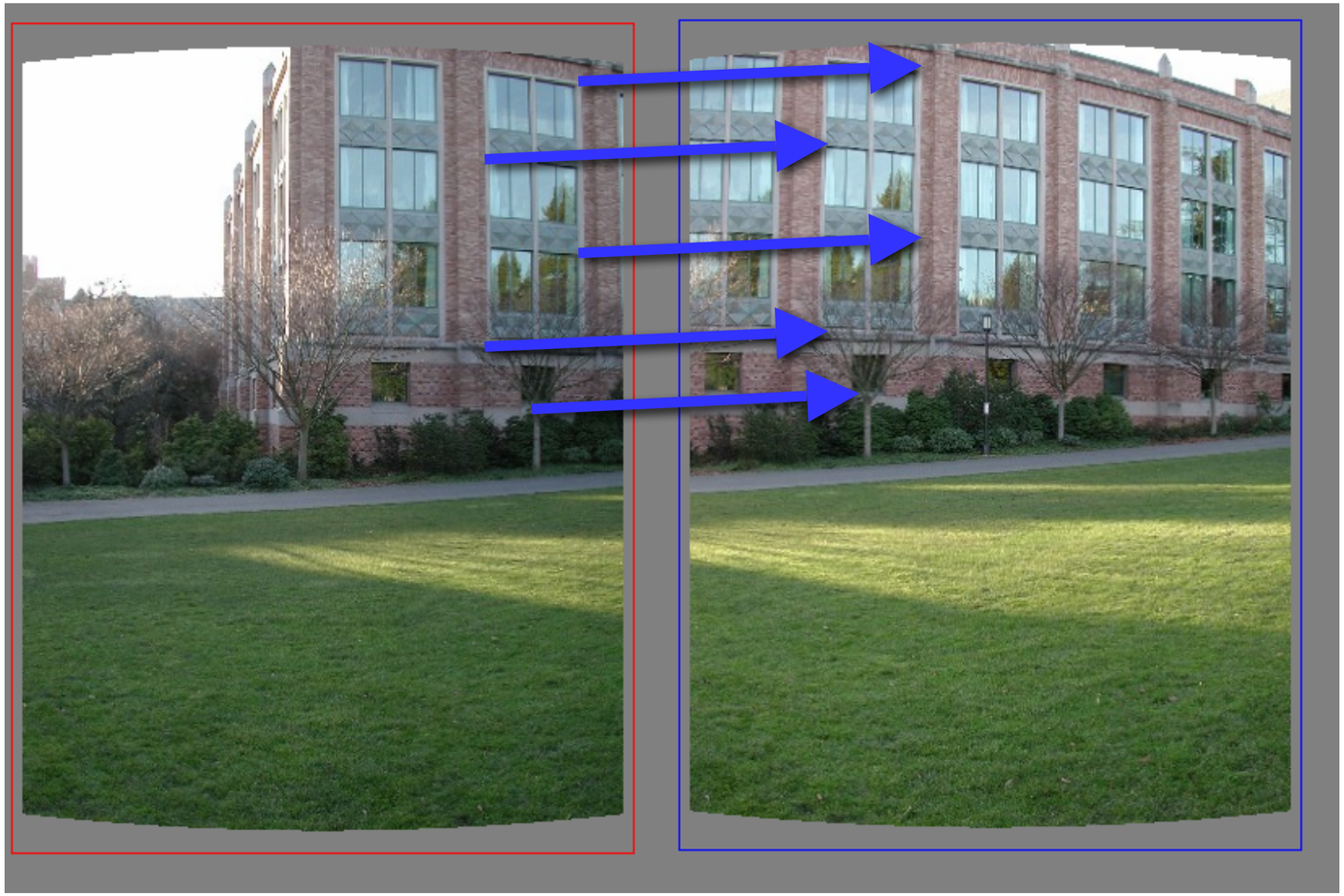
- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

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

Translations

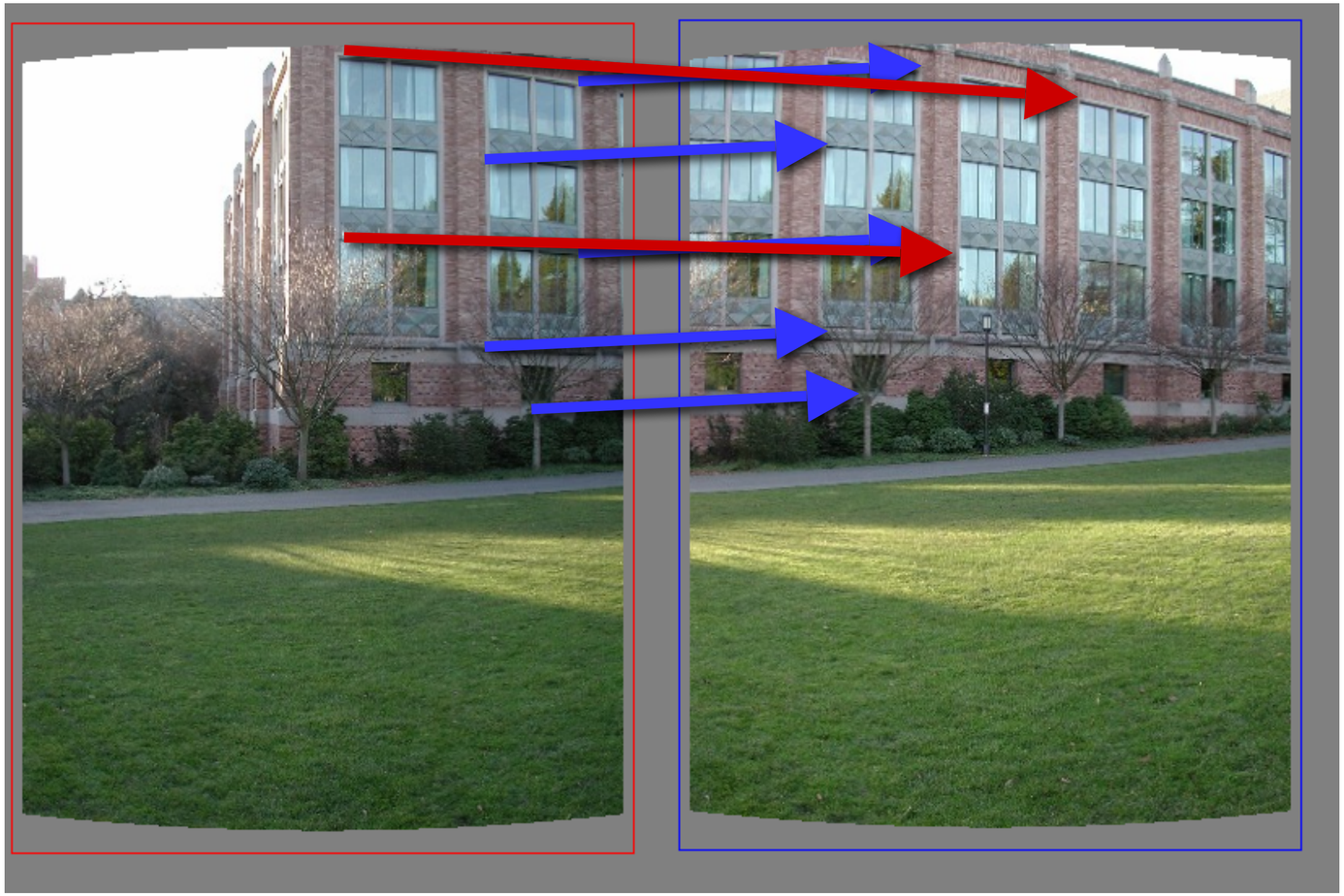
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



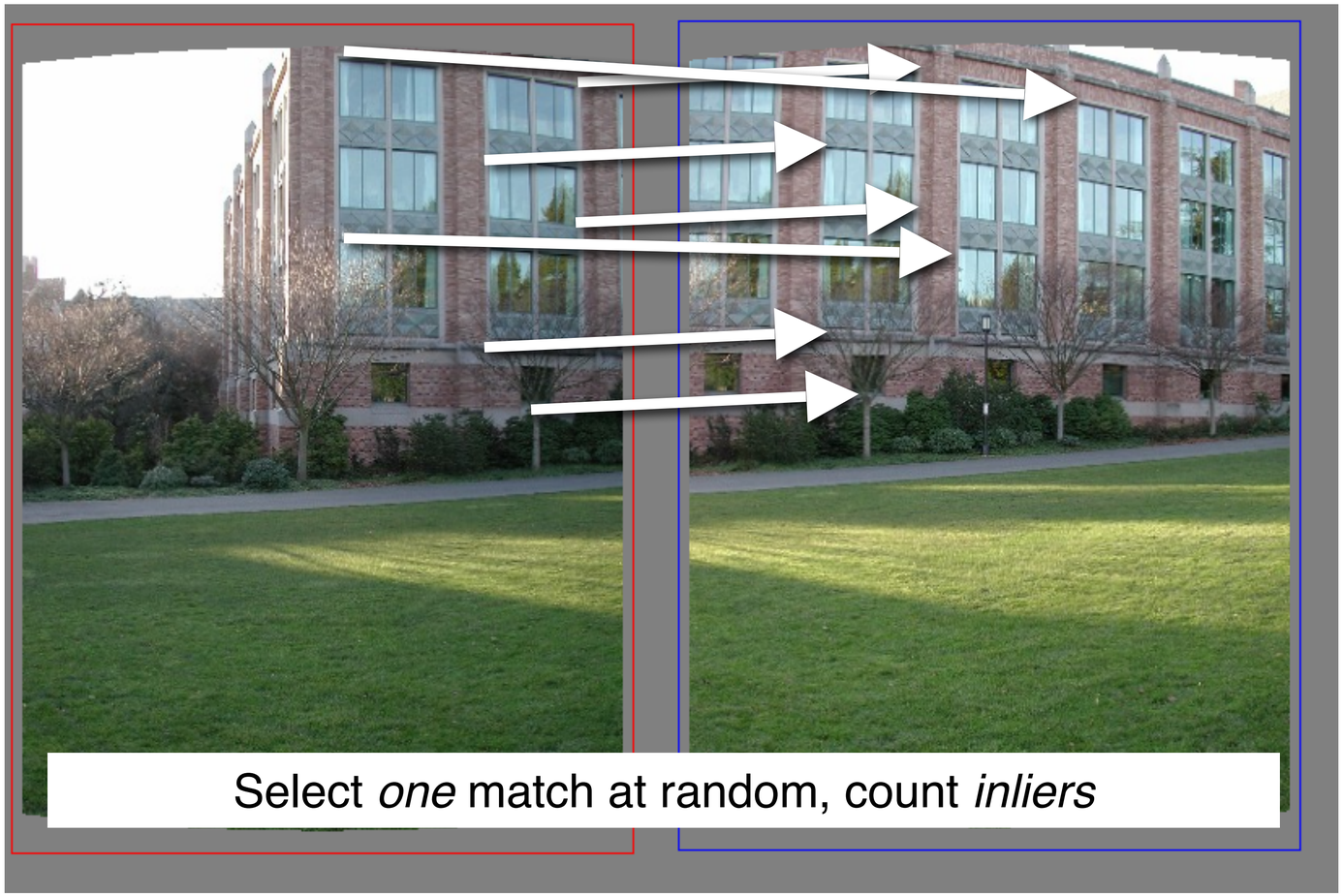
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

Translations



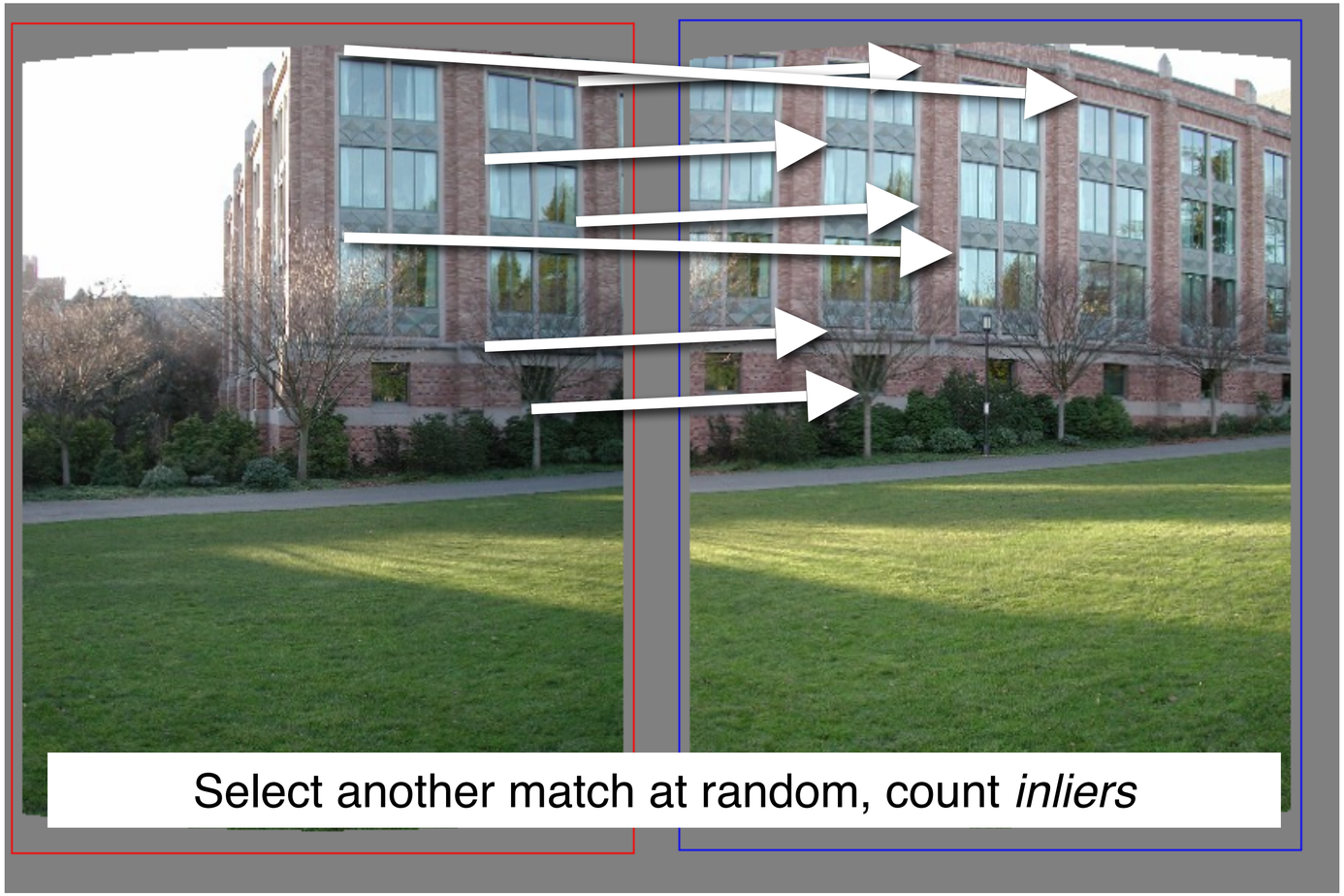
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RAndom SAmple Consensus



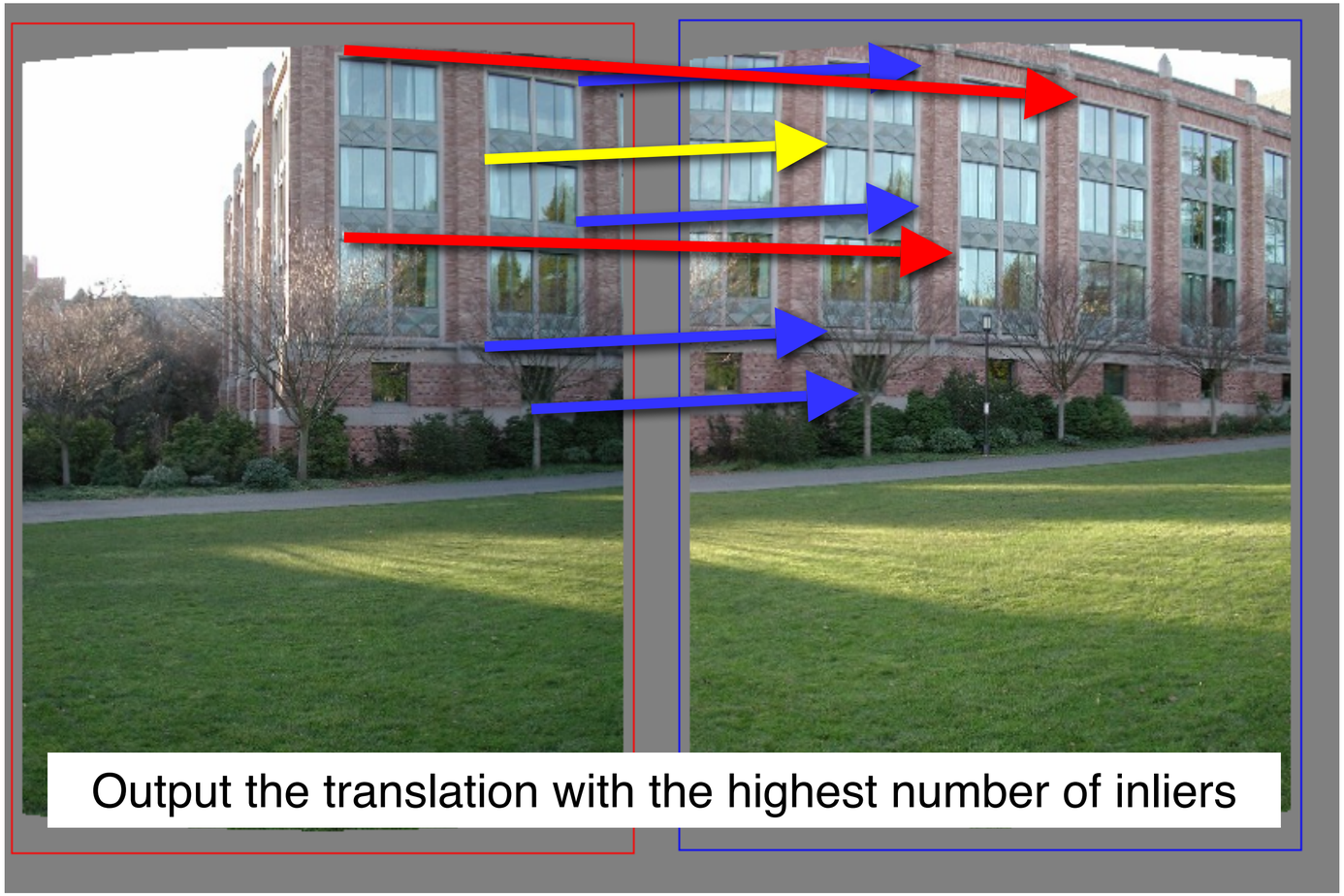
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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

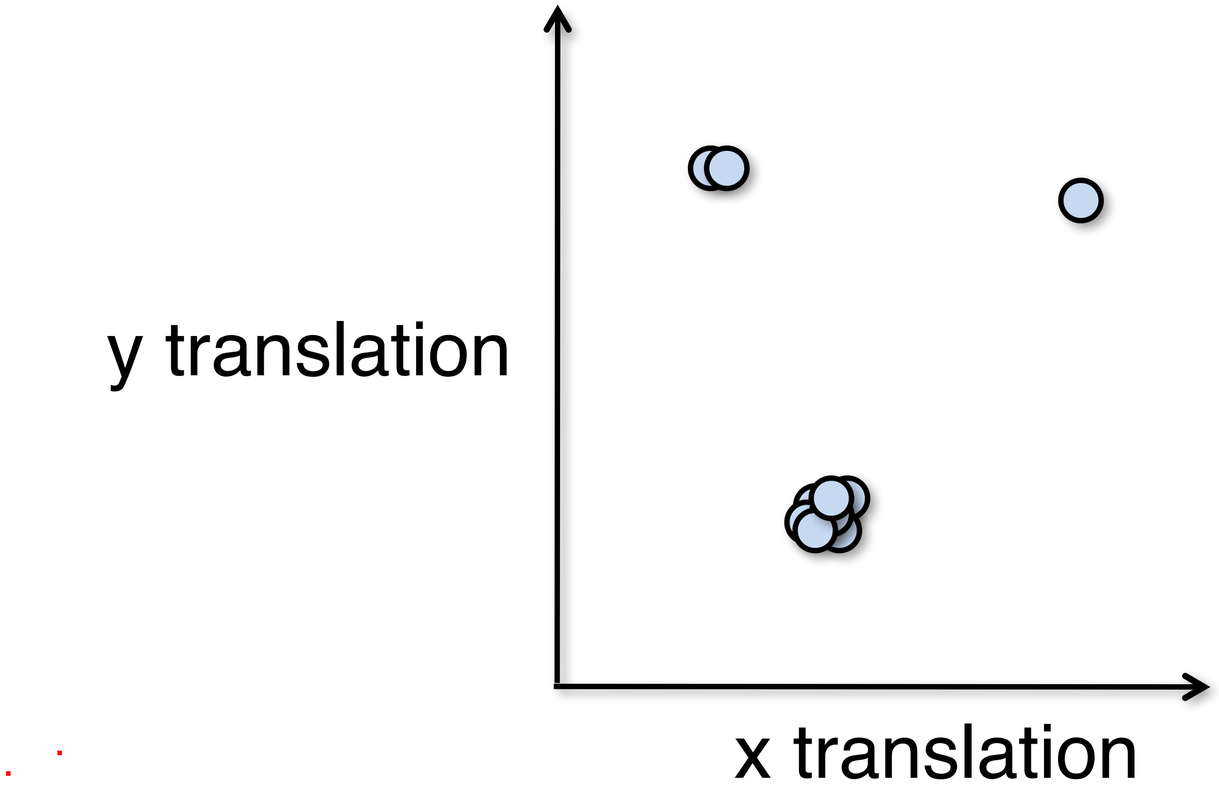
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

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?

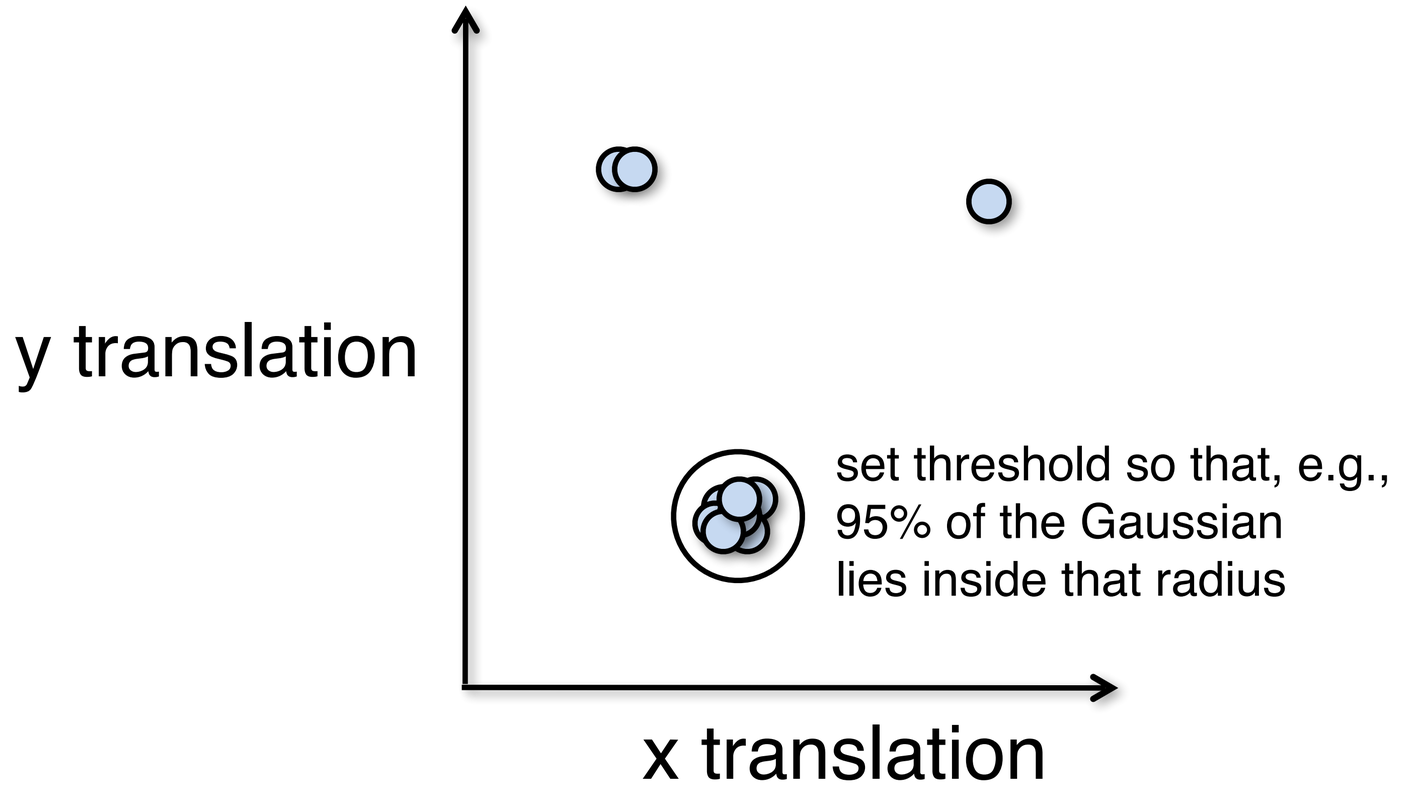
Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC: Another view



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

RANSAC: Another view

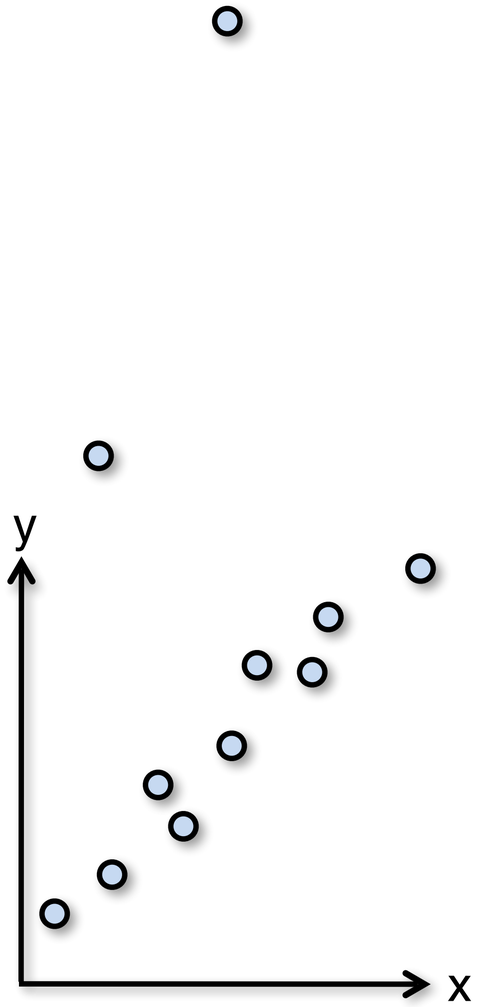


Slide credit: Noah Snavely, CS5670, https://www.cs.cornell.edu/courses/cs5670/2024sp/lectures/lectures.html

RANSAC

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

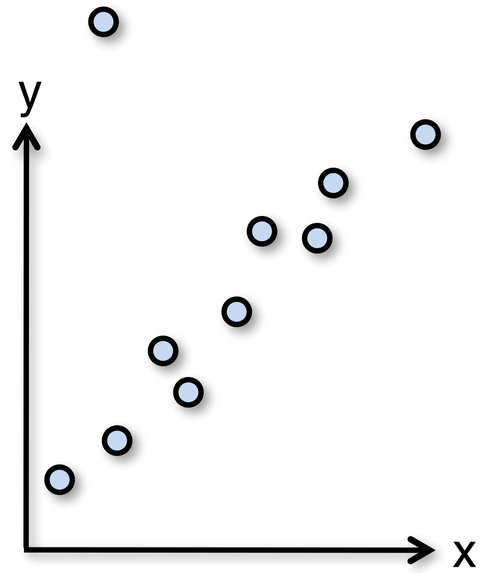
RANSAC



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

RANSAC

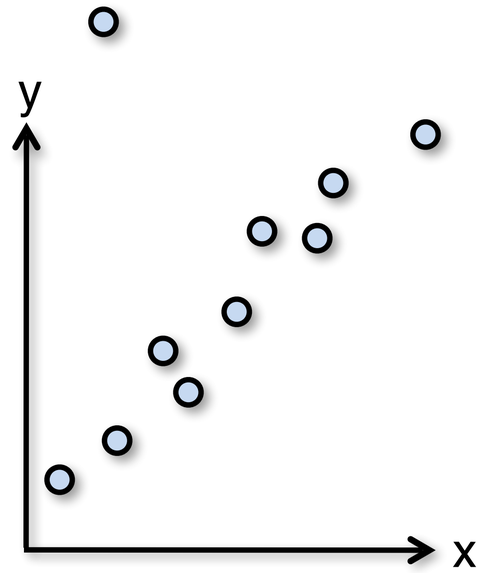
- Back to linear regression
- How do we generate a hypothesis?



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

RANSAC

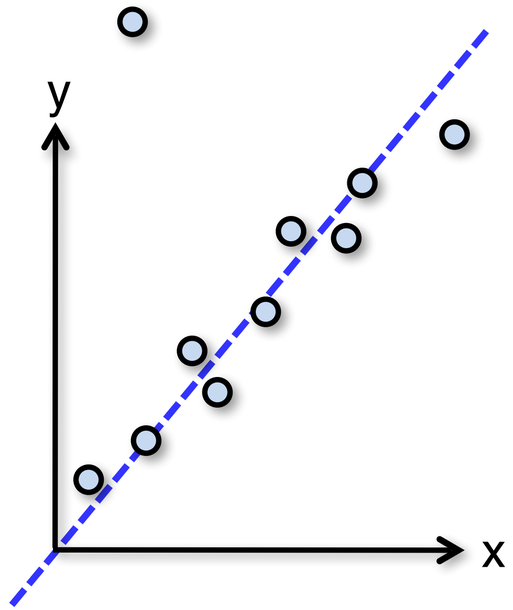
- Back to linear regression
- How do we generate a hypothesis?



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