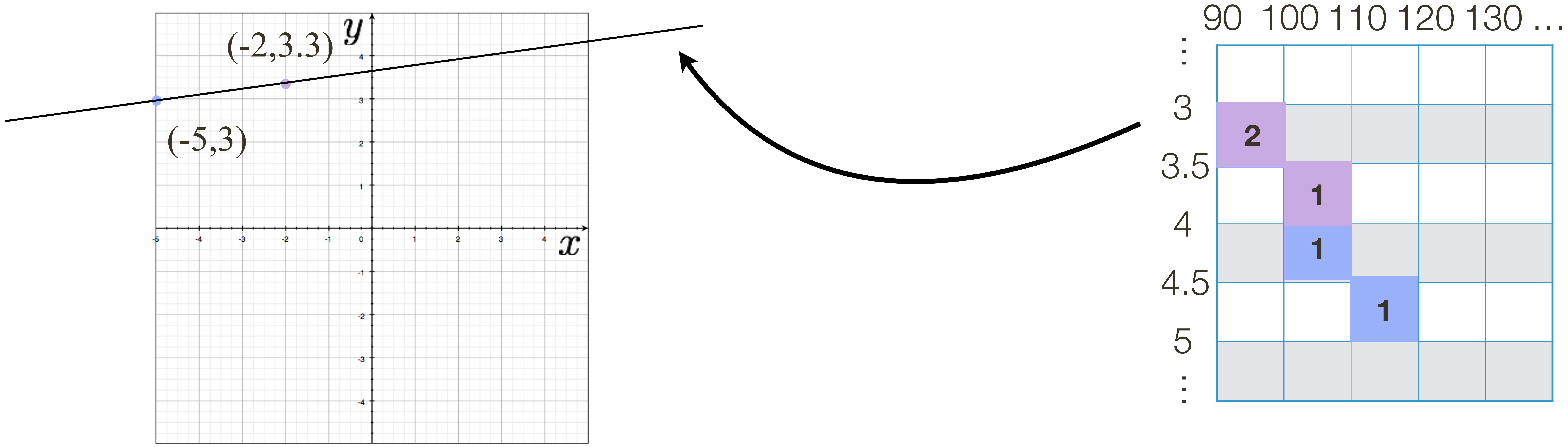
Bug fixed — Example: Hough Transform for Lines



$$-5\cos(95^\circ) + 3\cos(95^\circ) + r = 0 \rightarrow r \approx 3.42$$

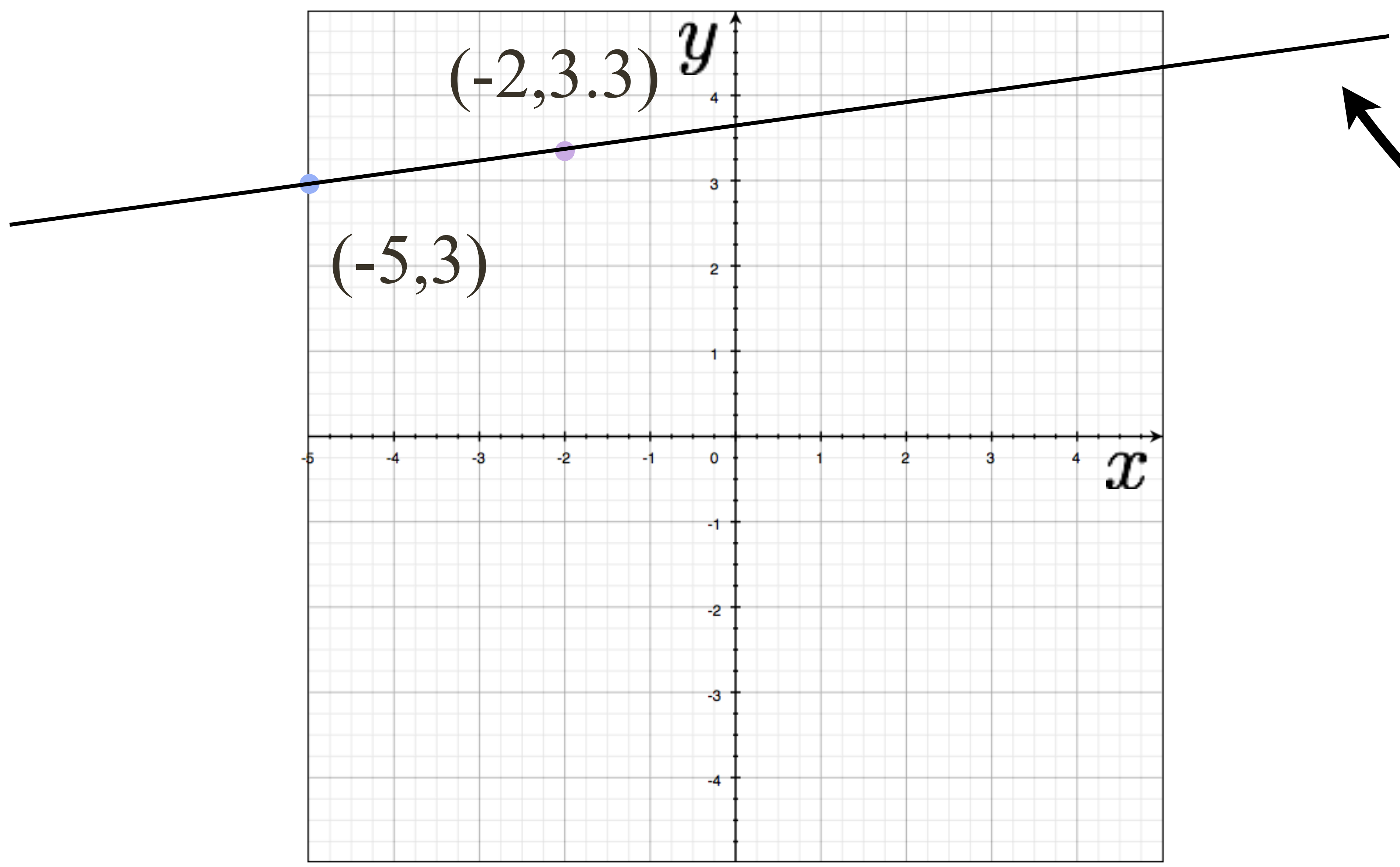
$$-5\cos(105^\circ) + 3\cos(105^\circ) + r = 0 \rightarrow r \approx 4.18$$

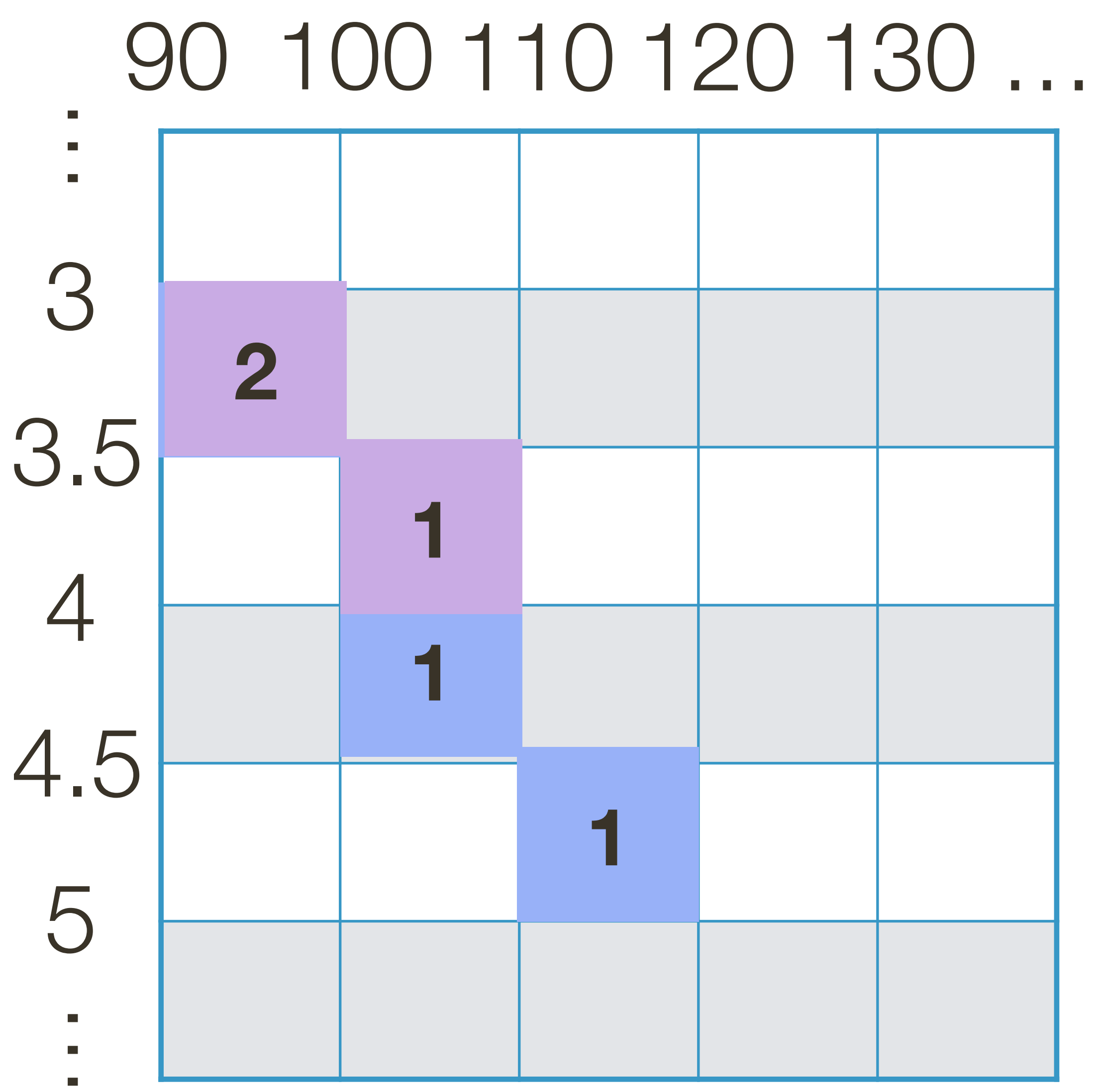
$$-5\cos(115^\circ) + 3\cos(115^\circ) + r = 0 \rightarrow r \approx 4.83$$

$$-2\cos(95^\circ) + 3.3\cos(95^\circ) + r = 0 \rightarrow r \approx 3.46$$

$$-2\cos(105^\circ) + 3.3\cos(105^\circ) + r = 0 \rightarrow r \approx 3.71$$

Bug really fixed — Example: Hough Transform for Lines





$$-5\cos(95^\circ) + 3\sin(95^\circ) + r = 0 \rightarrow r \approx 3.42$$

$$-5\cos(105^\circ) + 3\sin(105^\circ) + r = 0 \rightarrow r \approx 4.18$$

$$-5\cos(115^\circ) + 3\sin(115^\circ) + r = 0 \rightarrow r \approx 4.83$$

$$-2\cos(95^\circ) + 3.3\sin(95^\circ) + r = 0 \rightarrow r \approx 3.46$$

$$-2\cos(105^\circ) + 3.3\sin(105^\circ) + r = 0 \rightarrow r \approx 3.71$$

Learning Goals for Optical Flow

LINEARIZE

how do we find more equations?

Flow at a pixel

Look at previous equation at a single pixel:

$$\frac{\partial I_1}{\partial \mathbf{x}}^T \Delta \mathbf{u} = I_0(\mathbf{x}) - I_1(\mathbf{x})$$

 15.2

Optical Flow in **1D**

Consider a 1D function moving at velocity v

15.3

t

1	1	1	1	1
1	1	1	1	1
1	10	10	10	10
1	10	10	10	10
1	10	10	10	10
1	10	10	10	10

$t+1$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	10	10	10
1	1	10	10	10
1	1	10	10	10

$$I_x = \frac{\partial I}{\partial x}$$

-	0	0	0	-
-	0	0	0	-
-	9	0	0	-
-	9	0	0	-
-	9	0	0	-
-	9	0	0	-

-1 0 1

$$I_y = \frac{\partial I}{\partial y}$$

-	-	-	-	-
0	9	9	9	9
0	9	9	9	9
0	0	0	0	0
0	0	0	0	0
-	-	-	-	-

-1
0
1

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

0	0	0	0	0
0	0	0	0	0
0	-9	-9	-9	-9
0	-9	0	0	0
0	-9	0	0	0
0	-9	0	0	0

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

How do we **compute** …

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

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

spatial derivative

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

optical flow

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

temporal derivative

Forward difference
Sobel filter
Scharr filter
…

How do we solve for u and v?

Frame differencing

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Lucas-Kanade

Optical Flow Constraint Equation: $I_x u + I_y v + I_t = 0$

Suppose $[x_1, y_1] = [x, y]$ is the (original) center point in the **window**. Let $[x_2, y_2]$ be any other point in the window. This gives us two equations that we can write

$$I_{x_1} u + I_{y_1} v = -I_{t_1}$$
$$I_{x_2} u + I_{y_2} v = -I_{t_2}$$

and that can be solved locally for u and v as

$$\begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \end{bmatrix}^{-1} \begin{bmatrix} I_{t_1} \\ I_{t_2} \end{bmatrix}$$

provided that u and v are the same in both equations and provided that the required matrix inverse exists.

Lucas-Kanade

Optical Flow Constraint Equation: $I_x u + I_y v + I_t = 0$

Considering all n points in the **window**, one obtains

$$\begin{aligned} I_{x_1} u + I_{y_1} v &= -I_{t_1} \\ I_{x_2} u + I_{y_2} v &= -I_{t_2} \\ &\vdots \\ I_{x_n} u + I_{y_n} v &= -I_{t_n} \end{aligned}$$

which can be written as the matrix equation

$$\mathbf{A}\mathbf{v} = \mathbf{b}$$

where $\mathbf{v} = [u, v]^T$, $\mathbf{A} = \begin{bmatrix} I_{x_1} & I_{y_1} \\ I_{x_2} & I_{y_2} \\ \vdots & \vdots \\ I_{x_n} & I_{y_n} \end{bmatrix}$ and $\mathbf{b} = -\begin{bmatrix} I_{t_1} \\ I_{t_2} \\ \vdots \\ I_{t_n} \end{bmatrix}$

Lucas-Kanade

The standard least squares solution is

$$\bar{\mathbf{v}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Note that we can explicitly write down an expression for $\mathbf{A}^T\mathbf{A}$ as

$$\mathbf{A}^T\mathbf{A} = \left[\begin{array}{cc} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{array} \right]$$

Where have we seen this before?

Can this tell us something about where LK is likely to work well?

Lucas-Kanade **Summary**

A dense method to compute motion, $[u, v]$, at every location in an image

Key Assumptions:

1. Motion is slow enough and smooth enough that differential methods apply (i.e., that the partial derivatives, I_x, I_y, I_t, are well-defined)

2. The optical flow constraint equation holds (i.e., $\frac{dI(x,y,t)}{dt} = 0$)

3. A window size is chosen so that motion, $[u, v]$, is constant in the window

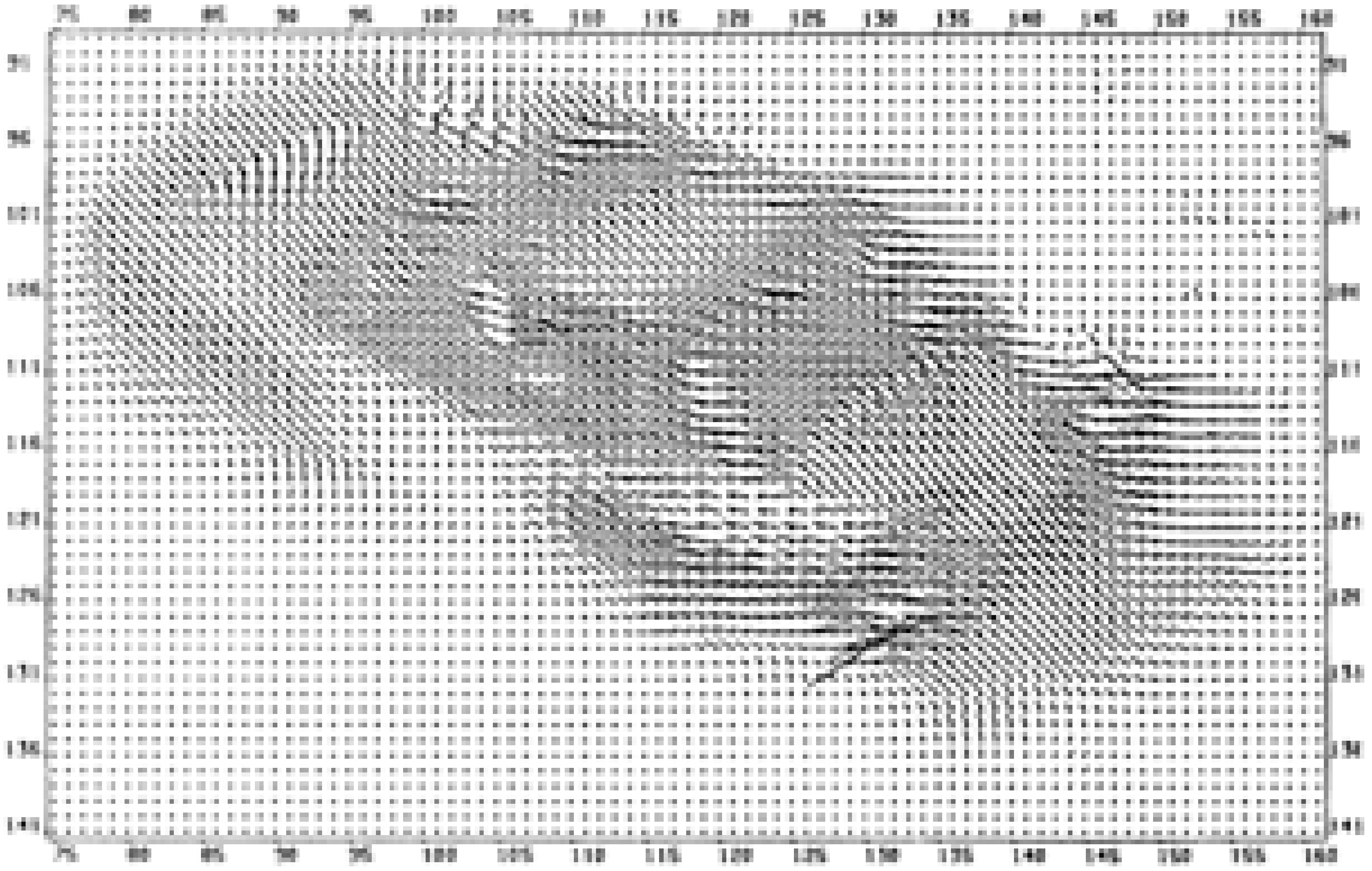
4. Windows are chosen s.t. that the rank of $\mathbf{A}^T\mathbf{A}$ is 2

Optical Flow **Smoothness Priors**

The optical flow equation gives **one constraint per pixel,** but we need to solve for 2 parameters u, v

Lucas Kanade adds constraints by **adding more pixels**

An alternative approach is to make assumptions about the **smoothness of the flow field**, e.g., that there should not be abrupt changes in flow



Optical Flow **Smoothness Priors**

Many methods trade off a ‘departure from the optical flow constraint’ cost with a ‘departure from smoothness’ cost.

$$\min_{\boldsymbol{u},\boldsymbol{v}} \sum_{i,j} \left\{ \overbrace{E_s(i,j)}^{\text{smoothness}} + \overbrace{\underbrace{\lambda}_{\text{weight}} E_d(i,j)}^{\text{brightness constancy}} \right\}$$

e.g., the Horn Schunck objective function penalises the magnitude of velocity:

$$E = \int\int (I_x u + I_y v + I_t)^2 + \lambda(|| \bigtriangledown u||^2 + || \bigtriangledown v||^2)$$

[Horn Schunck 1981, Szeliski p395]

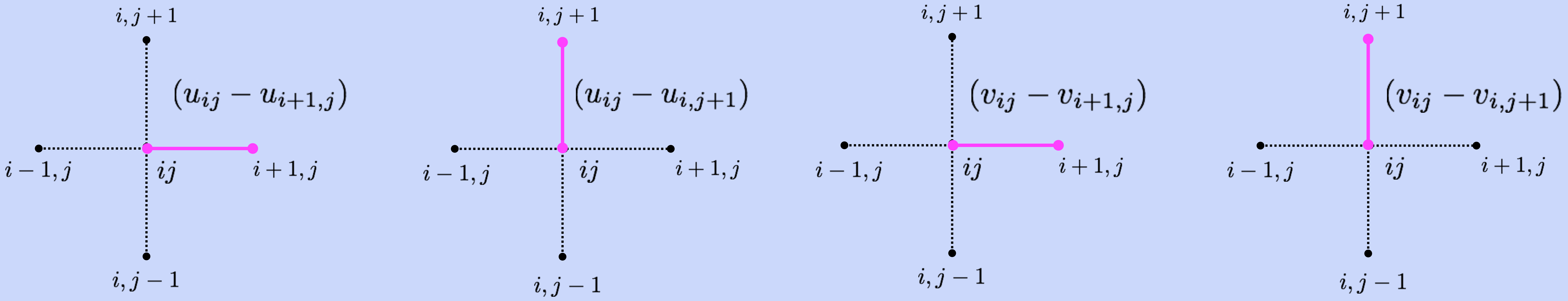
Horn-Schunck Optical Flow

Brightness constancy

$$E_d(i,j) = \left[I_x u_{ij} + I_y v_{ij} + I_t \right]^2$$

Smoothness

$$E_s(i,j) = \frac{1}{4}\left[(u_{ij} - u_{i+1,j})^2 + (u_{ij} - u_{i,j+1})^2 + (v_{ij} - v_{i+1,j})^2 + (v_{ij} - v_{i,j+1})^2 \right]$$



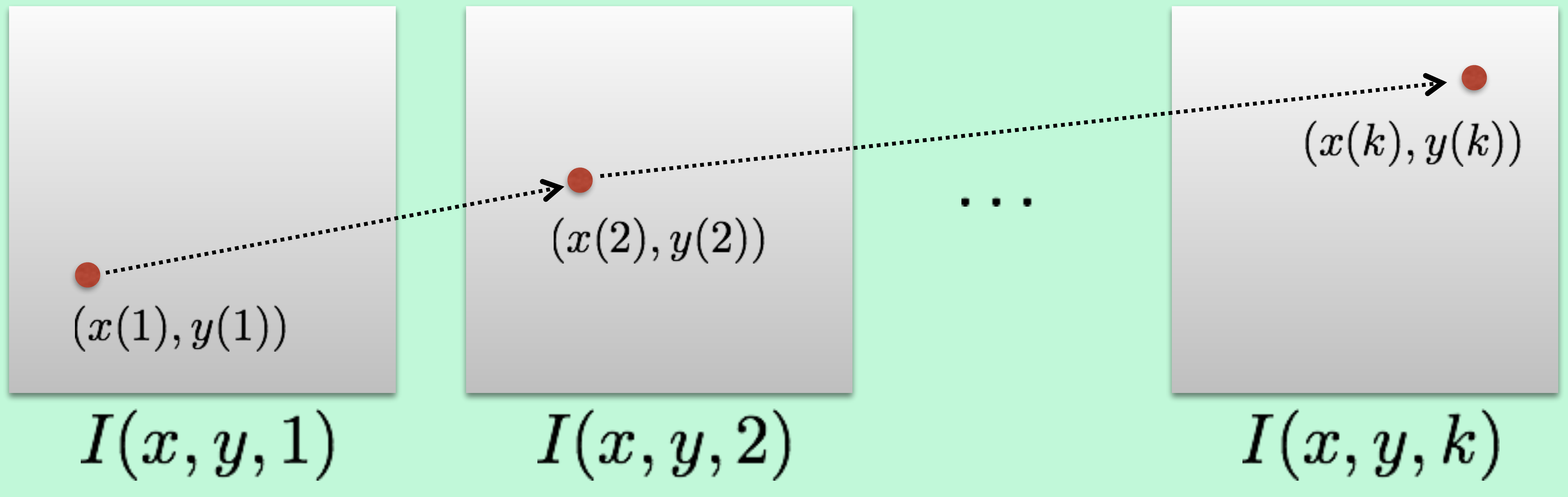
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Brightness **Constancy**

- All the methods presented in this lecture have relied on the assumption that
$$I_1(\mathbf{x} + \mathbf{u}) \approx I_0(\mathbf{x})$$
- This is called the **brightness constancy** assumption
- Taylor expansion for small motion at a single pixel → optical flow constraint
$$I_x u + I_y v + I_t = 0$$
- Horn-Schunk = optical flow constraint + smoothing over **u**
- Lucas-Kanade = optical flow constraint over patches assuming **u** is constant/slowly varying over patch

Optical Flow **Constraint Equation**

Brightness Constancy Assumption: Brightness of the point remains the same



$$I(x(t), y(t), t) = C$$

constant

15.4 What does this mean, and why is it reasonable?

Suppose $\dfrac{dI(x,y,t)}{dt} = 0$. Then we obtain the (classic) **optical flow constraint equation**

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

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Optical Flow and 2D Motion

Motion is geometric, **Optical flow** is radiometric

Usually we assume that optical flow and 2-D motion coincide ... but this is not always the case!

Optical flow with **no motion:**

. . . moving light source(s), lights going on/off, inter-reflection, shadows

Motion with **no optical flow:**

. . . spinning cylinder, sphere.

Optical Flow **Summary**

Motion, like binocular stereo, can be formulated as a matching problem. That is, given a scene point located at (x_0, y_0) in an image acquired at time t_0, what is its position, (x_1, y_1), in an image acquired at time t_1?

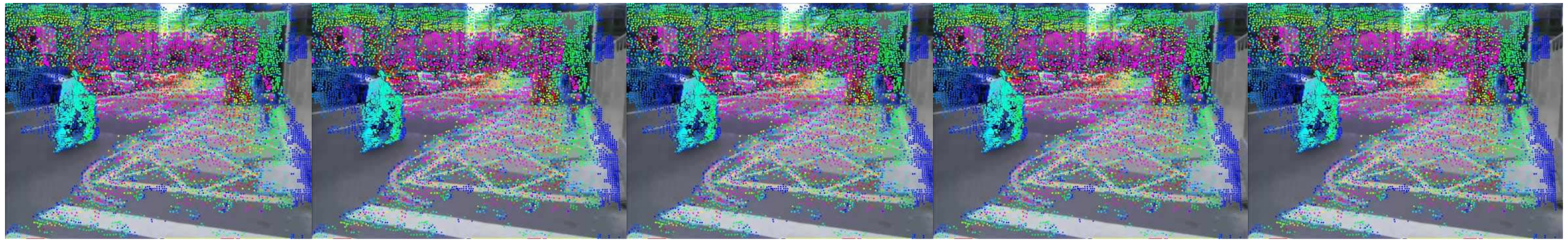
Assuming image intensity does not change as a consequence of motion, we obtain the (classic) **optical flow constraint equation**

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

where $[u, v]$, is the 2-D motion at a given point, $[x, y]$, and I_x, I_y, I_t are the partial derivatives of intensity with respect to x, y, and t

Lucas–Kanade is a dense method to compute the motion, $[u, v]$, at every location in an image

CPSC 425: Computer Vision



Lecture 17: Multiview Reconstruction

Menu for Today

Topics:

— **Stereo, Optical Flow** recap
— **Multiview Reconstruction**

Readings:

— **Today's** Lecture: Szeliski 11.4, 12.3-12.4, 9.3

Reminders:

— **Assignment 4**: due **March 20th**
— **Assignment 5**: Scene Recognition with Bag of Words is now available

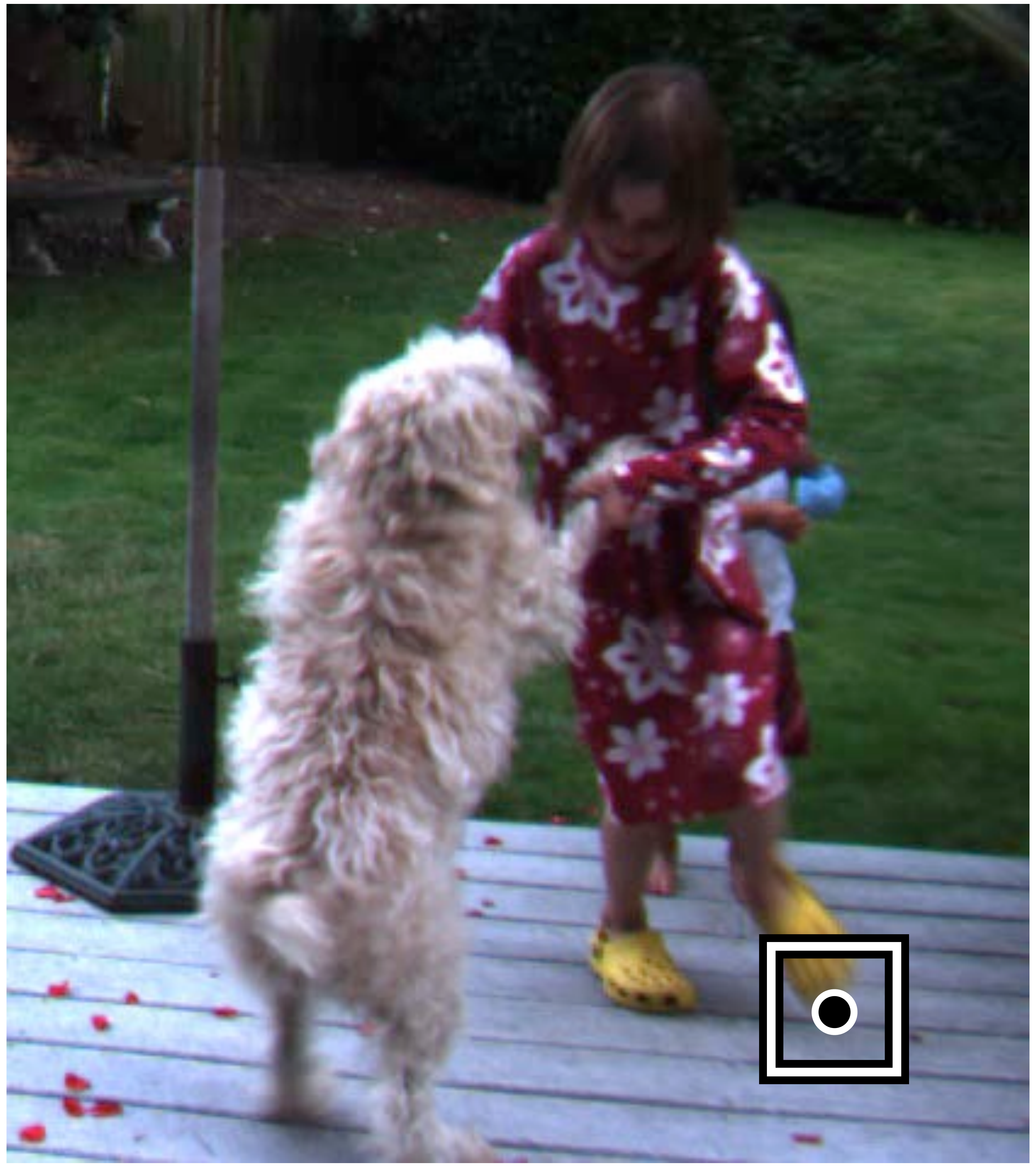
Putting it all together

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Non-Rigid** Matching

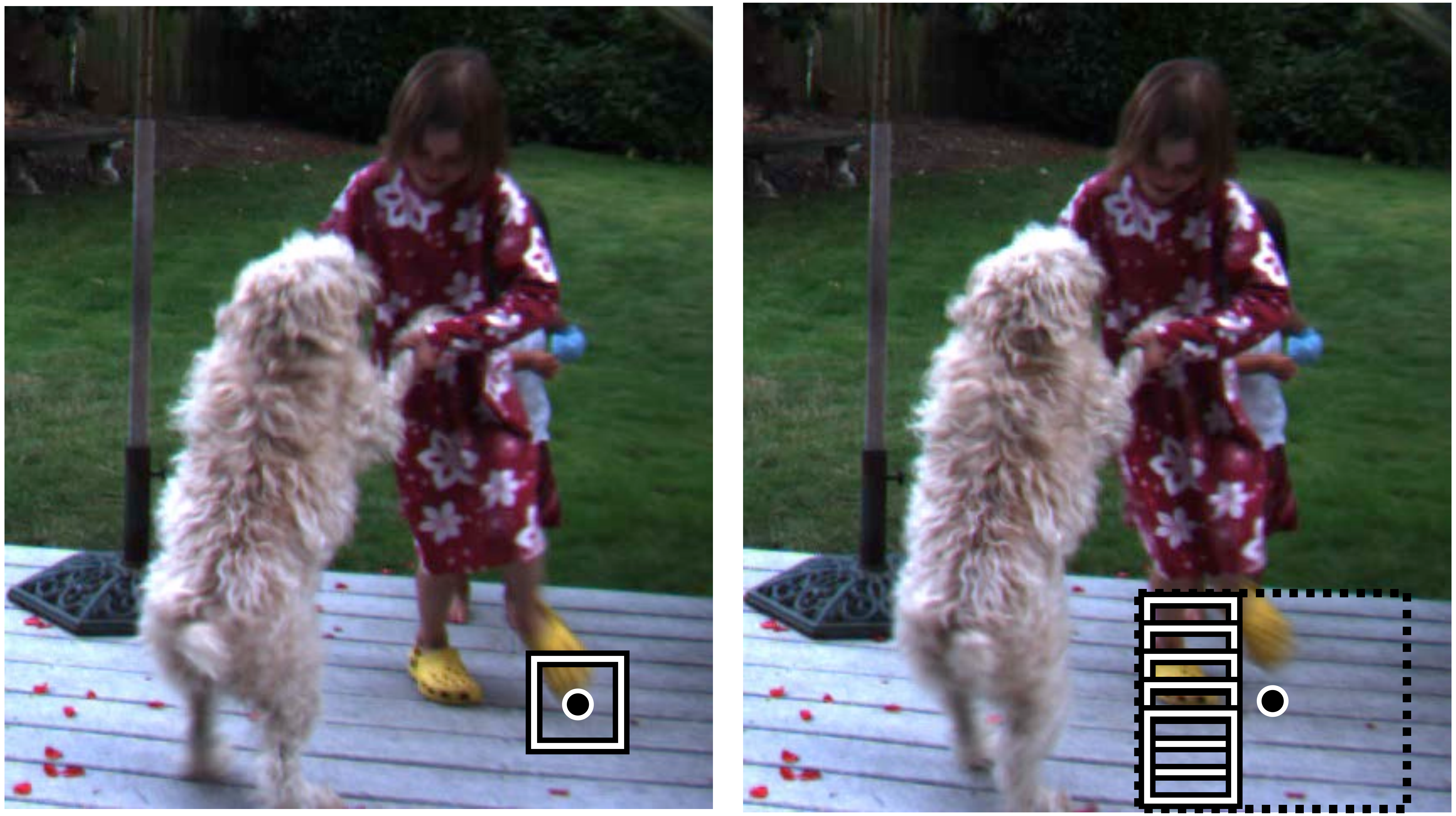
2D search, points can move anywhere in the image




[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

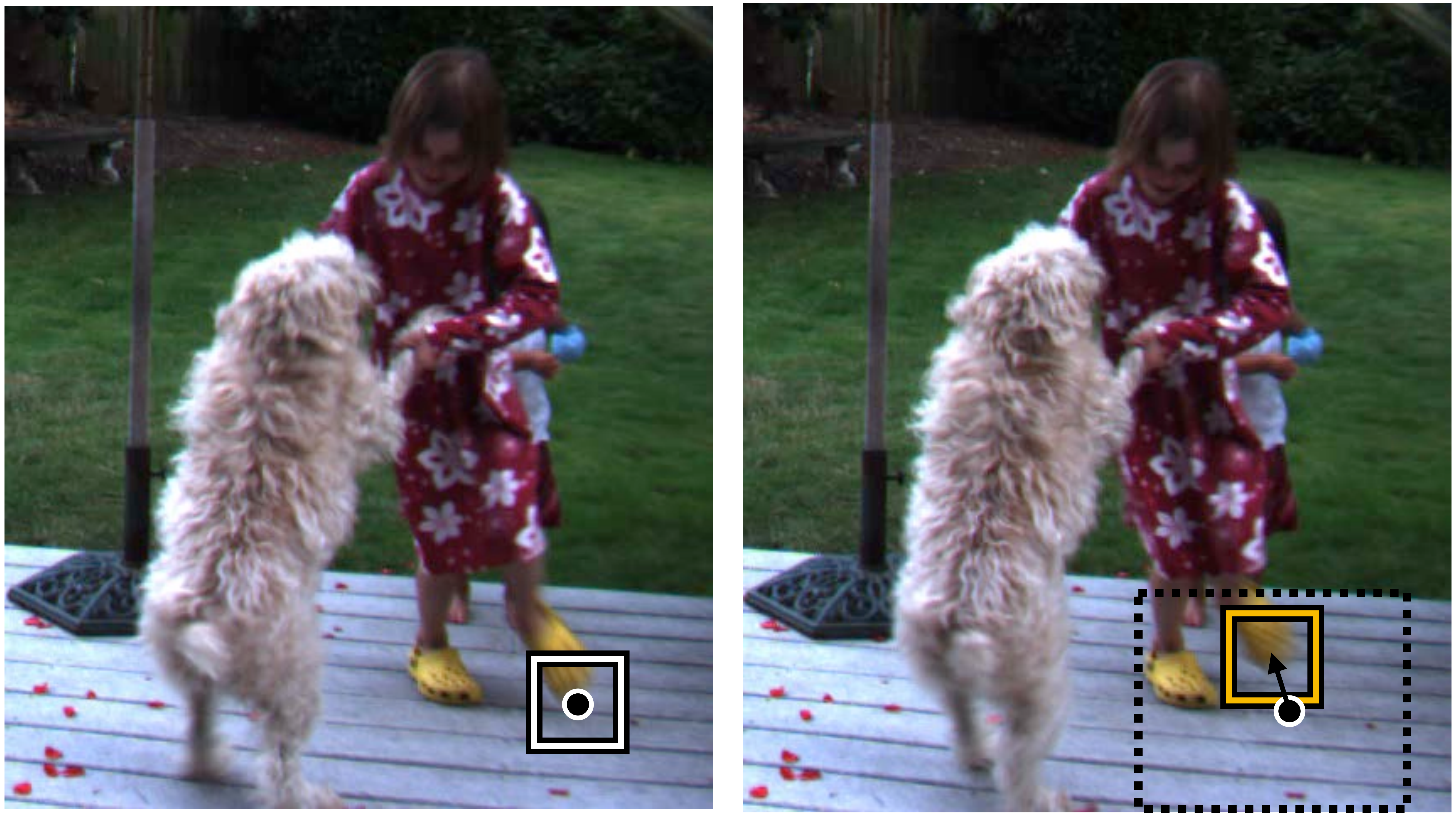
2D search, points can move anywhere in the image



[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

2D search, points can move anywhere in the image



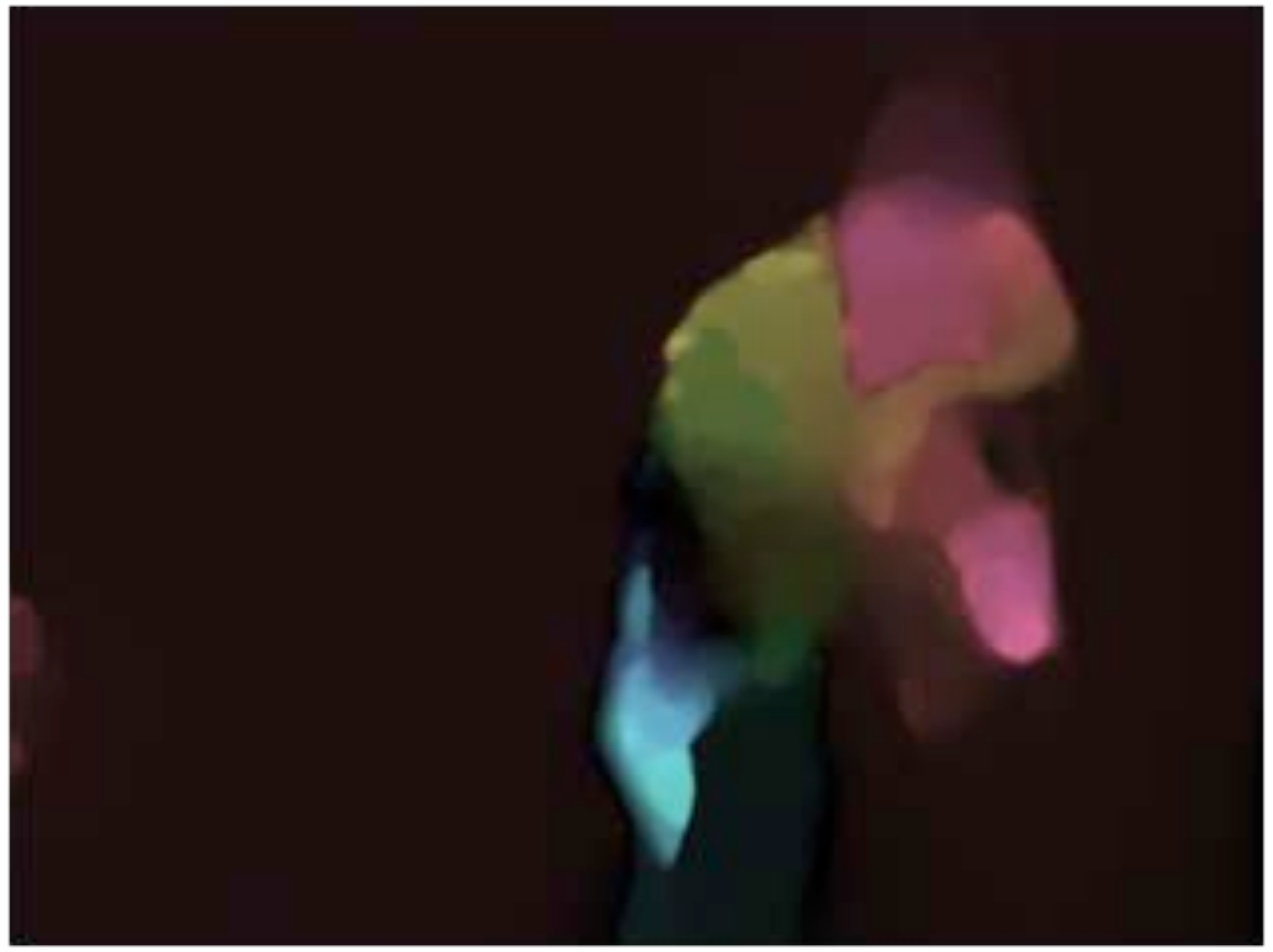
[vision.middlebury.edu/flow]

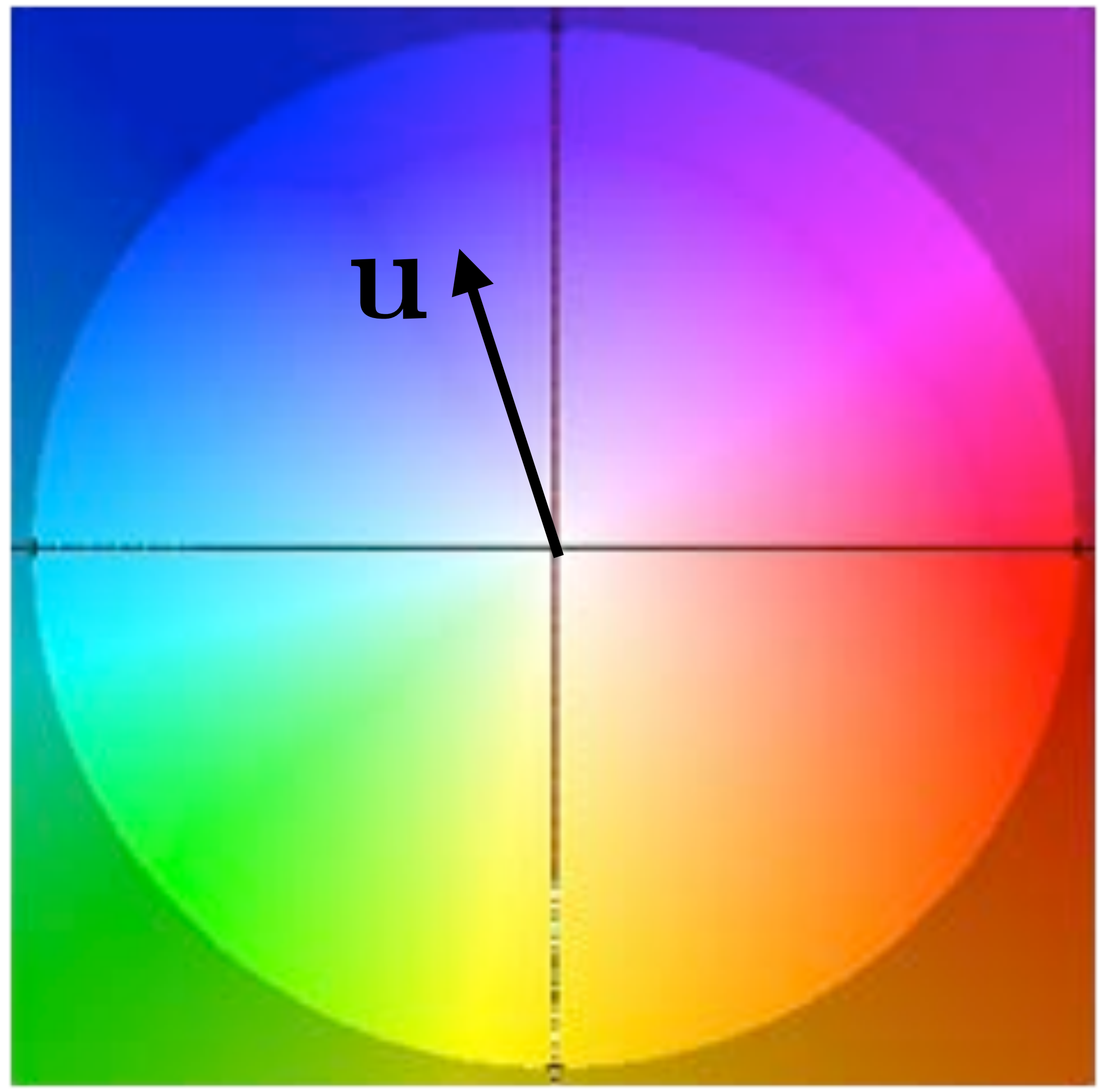
2-view **Non-Rigid** Matching

2D search, points can move anywhere in the image

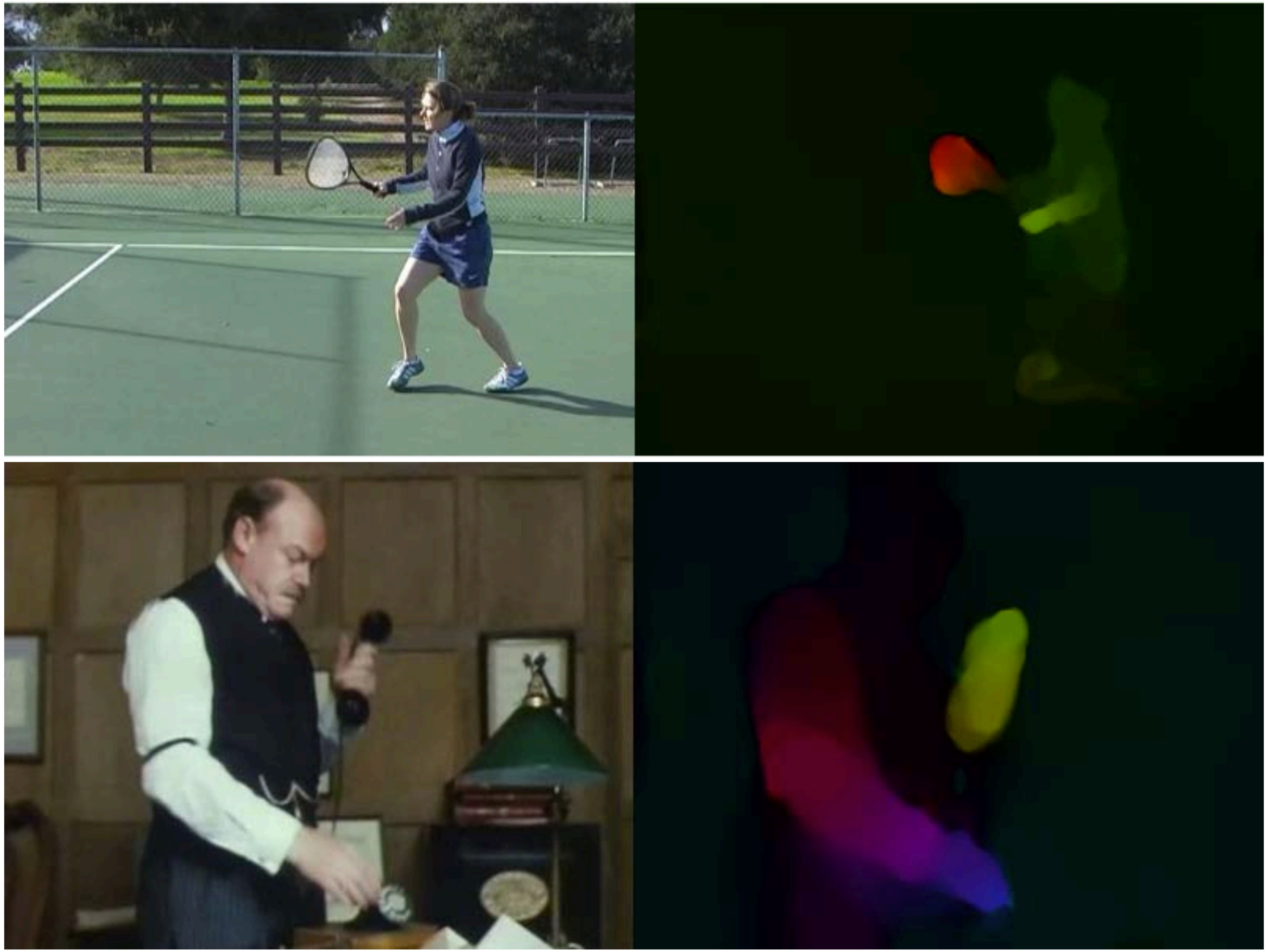
[vision.middlebury.edu/flow]

Optical Flow: Example 1





Optical Flow: Example 2



[Brox Malik 2011]

Optical Flow Recap

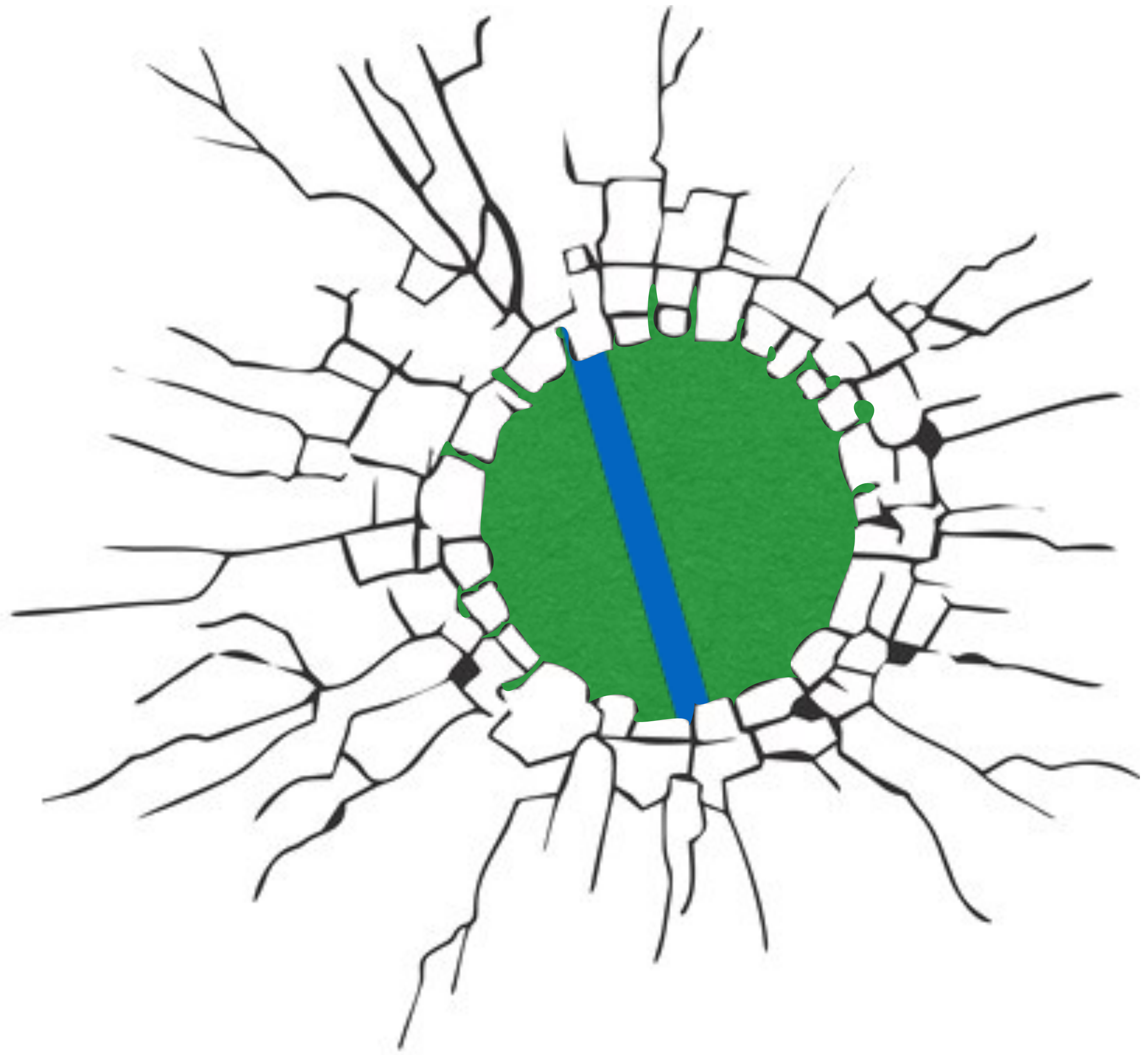
Optical Flow the apparent motion of all pixels in an image between a pair of image frames

Brightness Constancy a point on an object has the same intensity as it moves (in x, y, t)

Optical Flow Constraint the derivative of brightness constancy at a point, relates image gradients in x, y, t and flow vector u,v

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

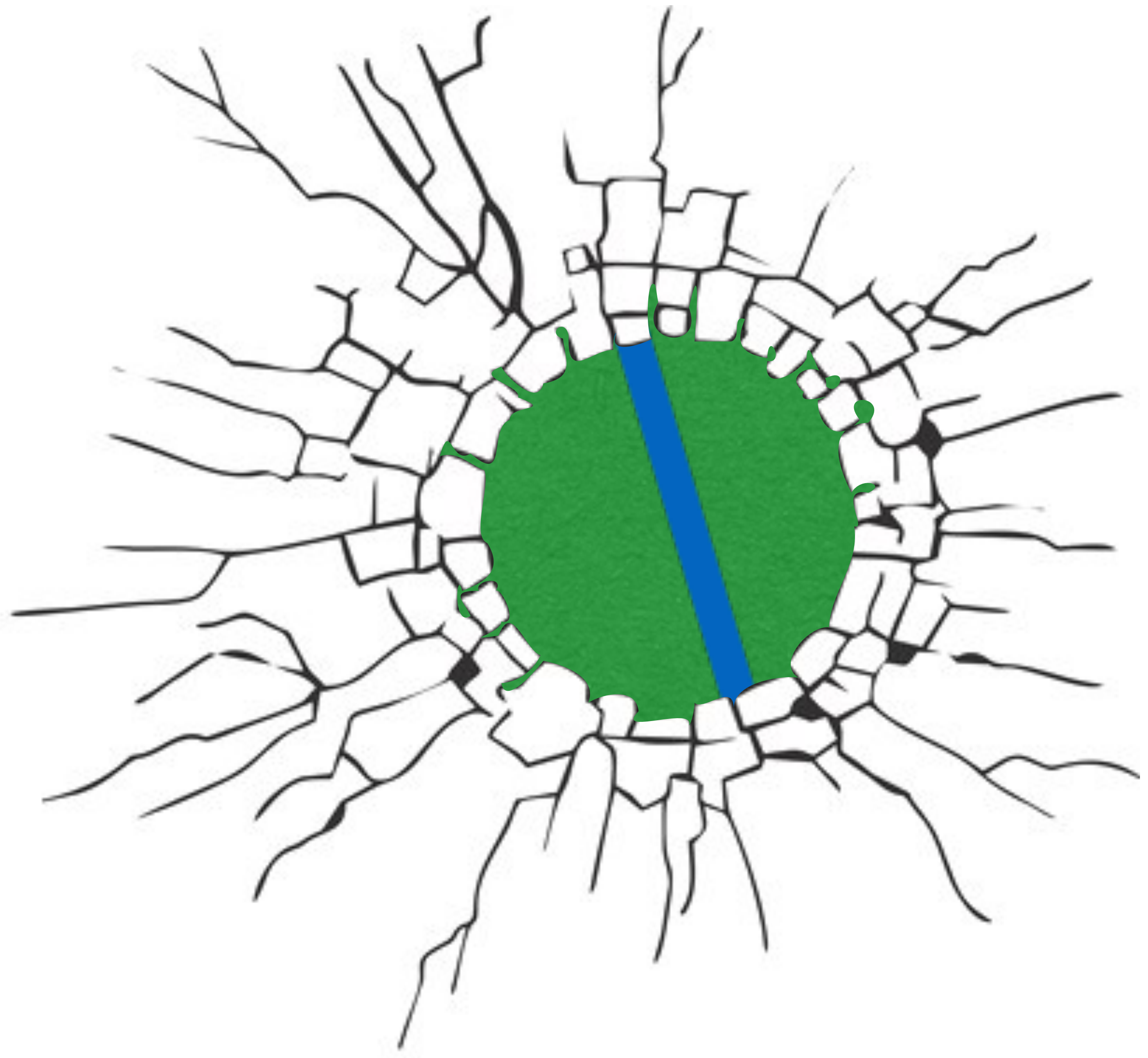
Aperture Problem



In which direction is the line moving?

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem



In which direction is the line moving?

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

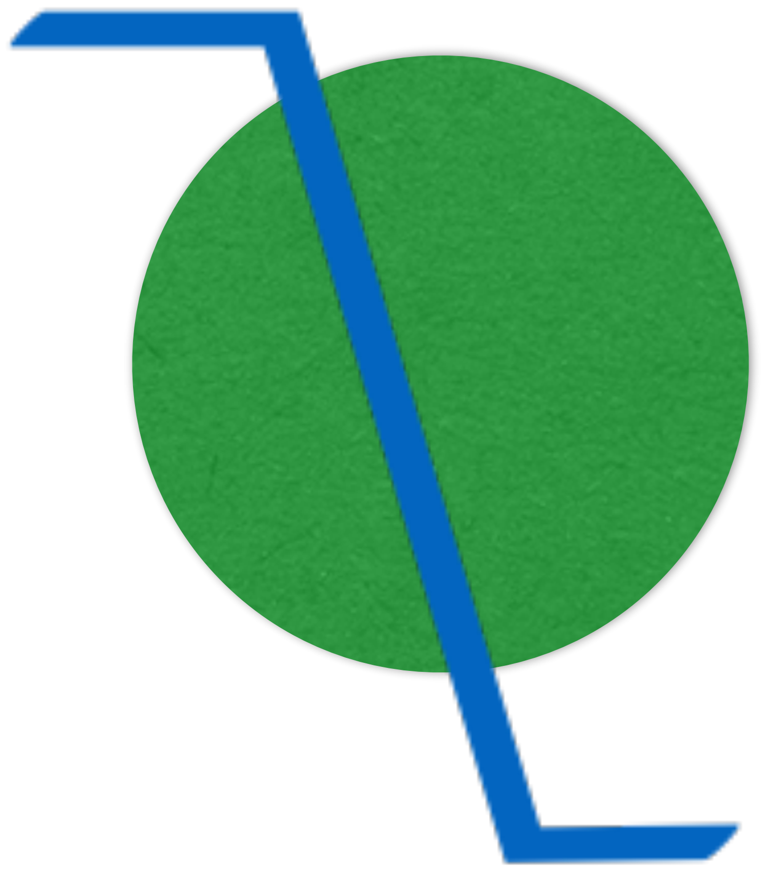


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

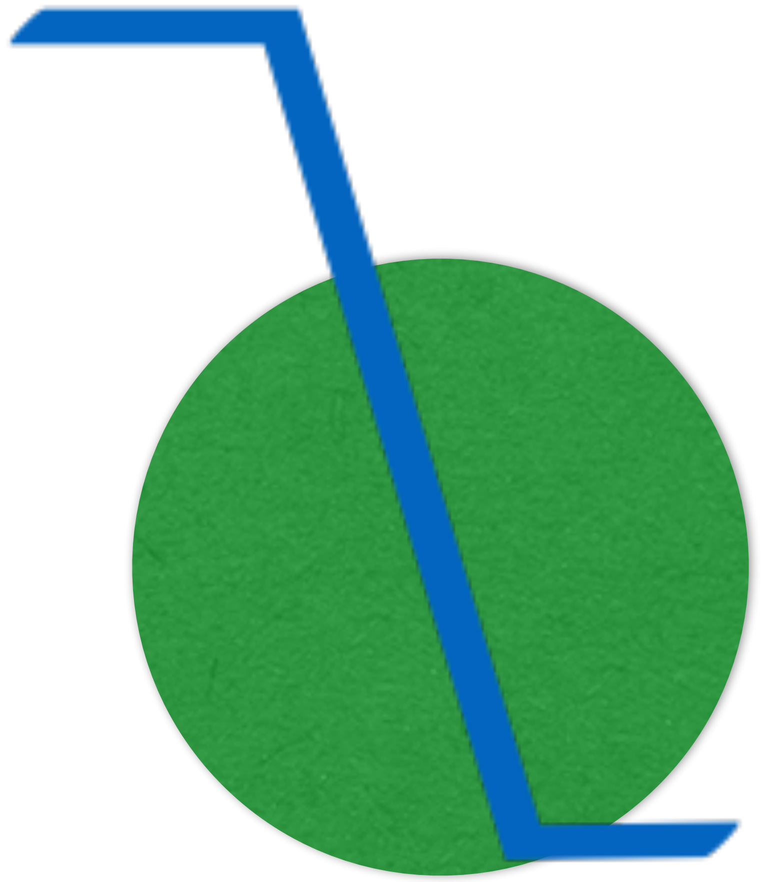


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

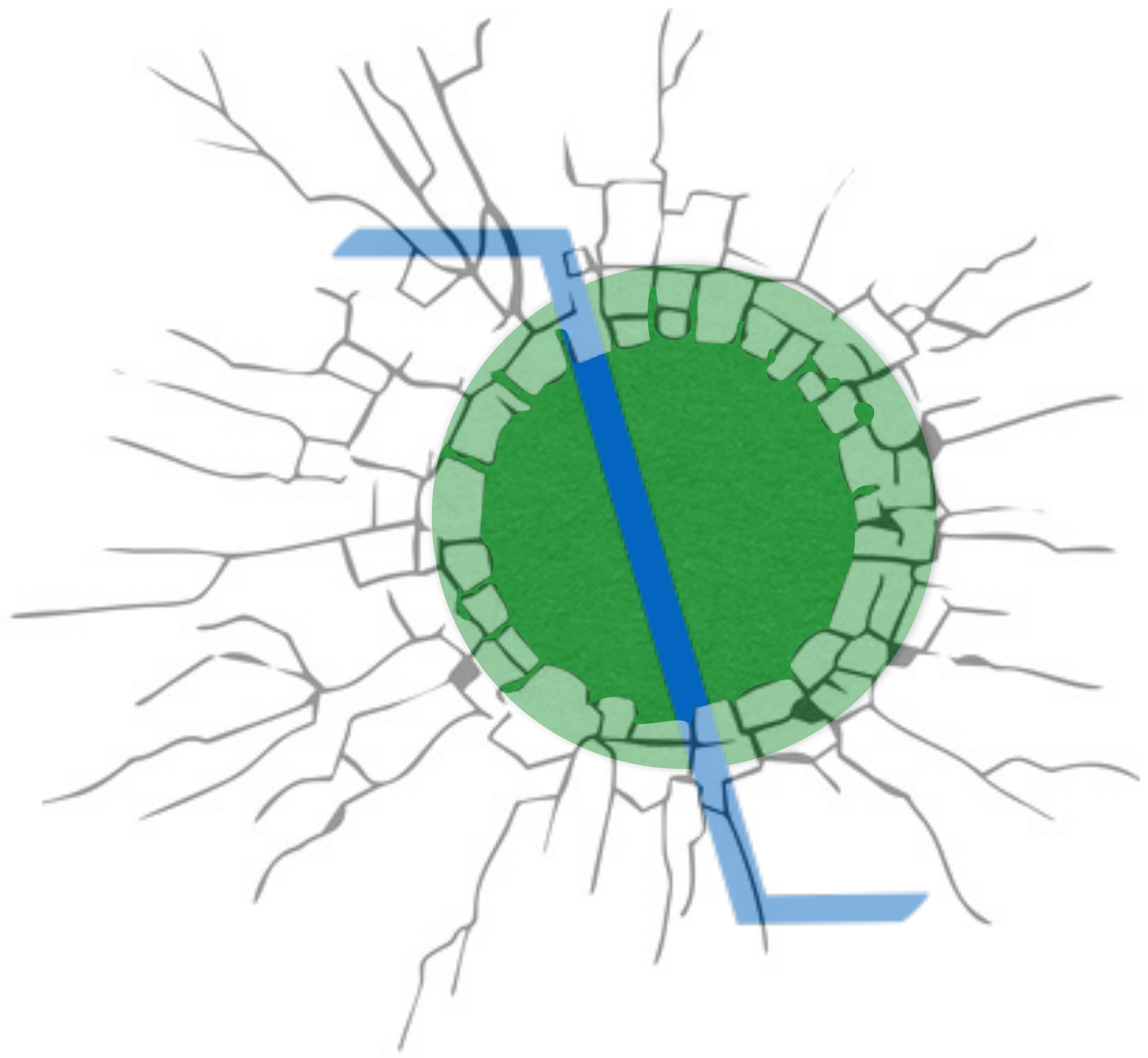


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

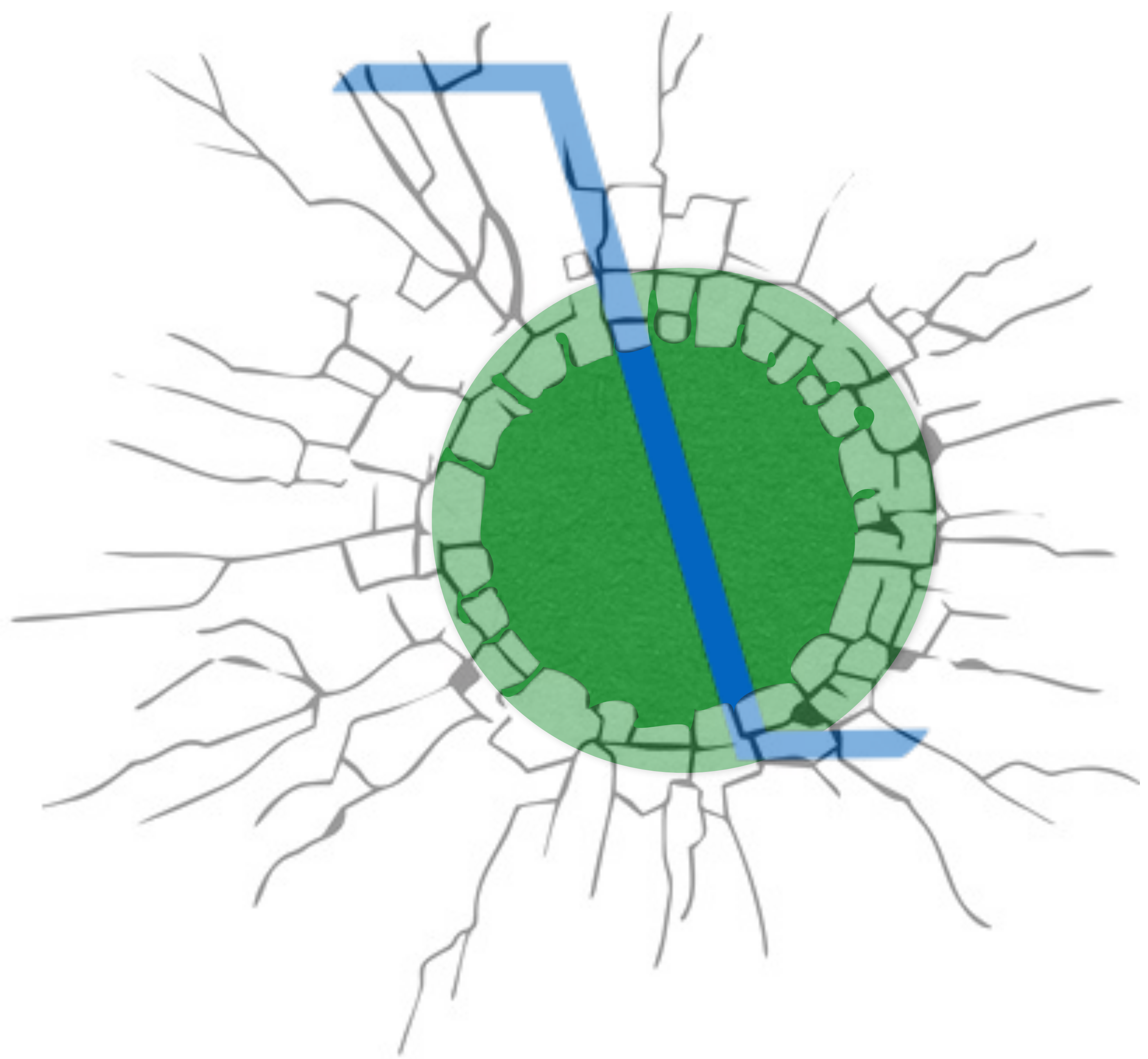


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Optical Flow Algorithms

Flow Ambiguity component of velocity parallel to an edge is not defined by the above constraint → aperture problem

Lucas Kanade resolves the ambiguity by adding multiple pixels (each with 1D optical flow constraint) → linear system to solve for 2D flow

Other flow algorithms (e.g., Horn Schunck) use **priors over the 2D flow field** (e.g., smoothness)

Multiview + Sparse SFM

- Multiview Image Alignment, Residuals, Error Function
- Structure from Motion (SFM)
- Bundle Adjustment, Pose Estimation, Triangulation

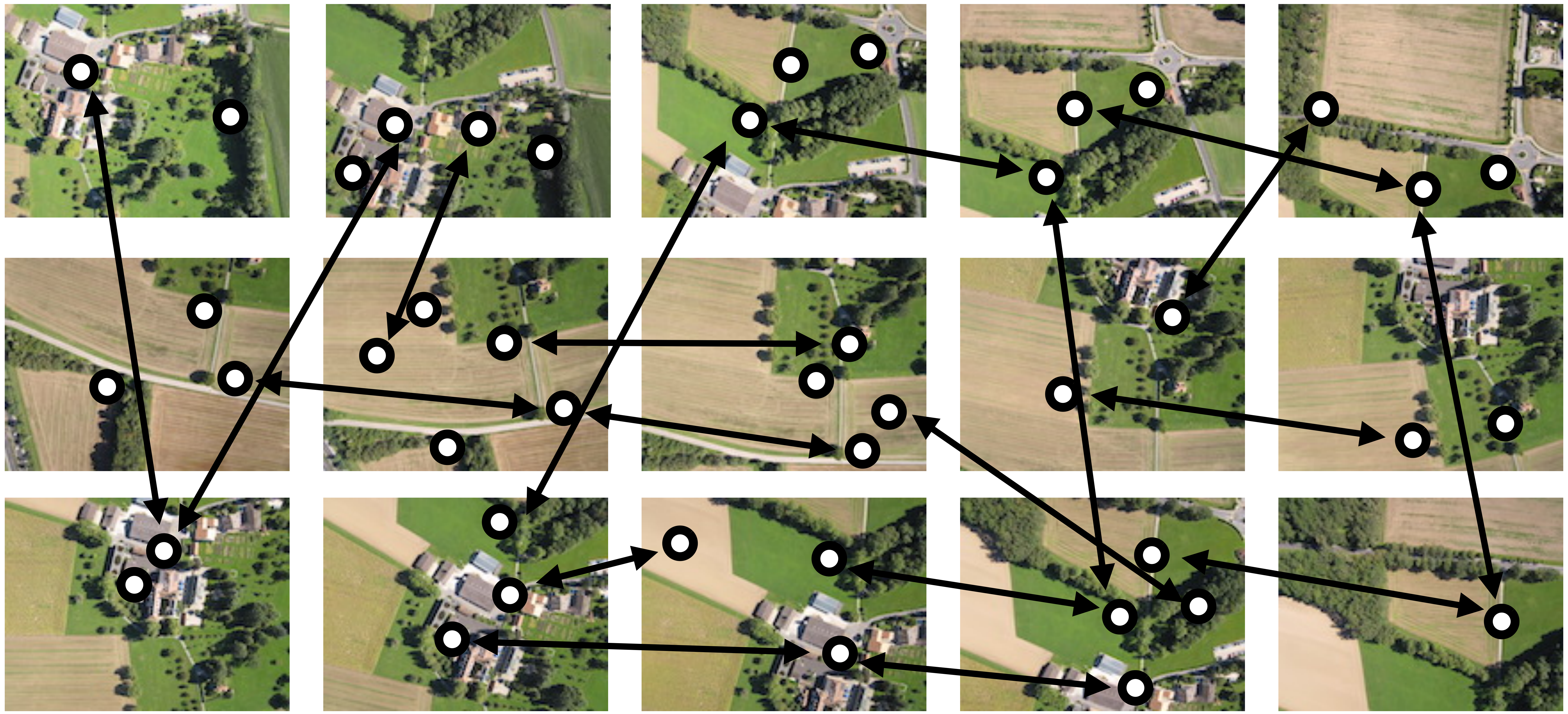
[Szeliski 11.4]

Multiview Image Alignment

Align a set of images given a motion model (e.g., planar affine)

Multiview Image Alignment

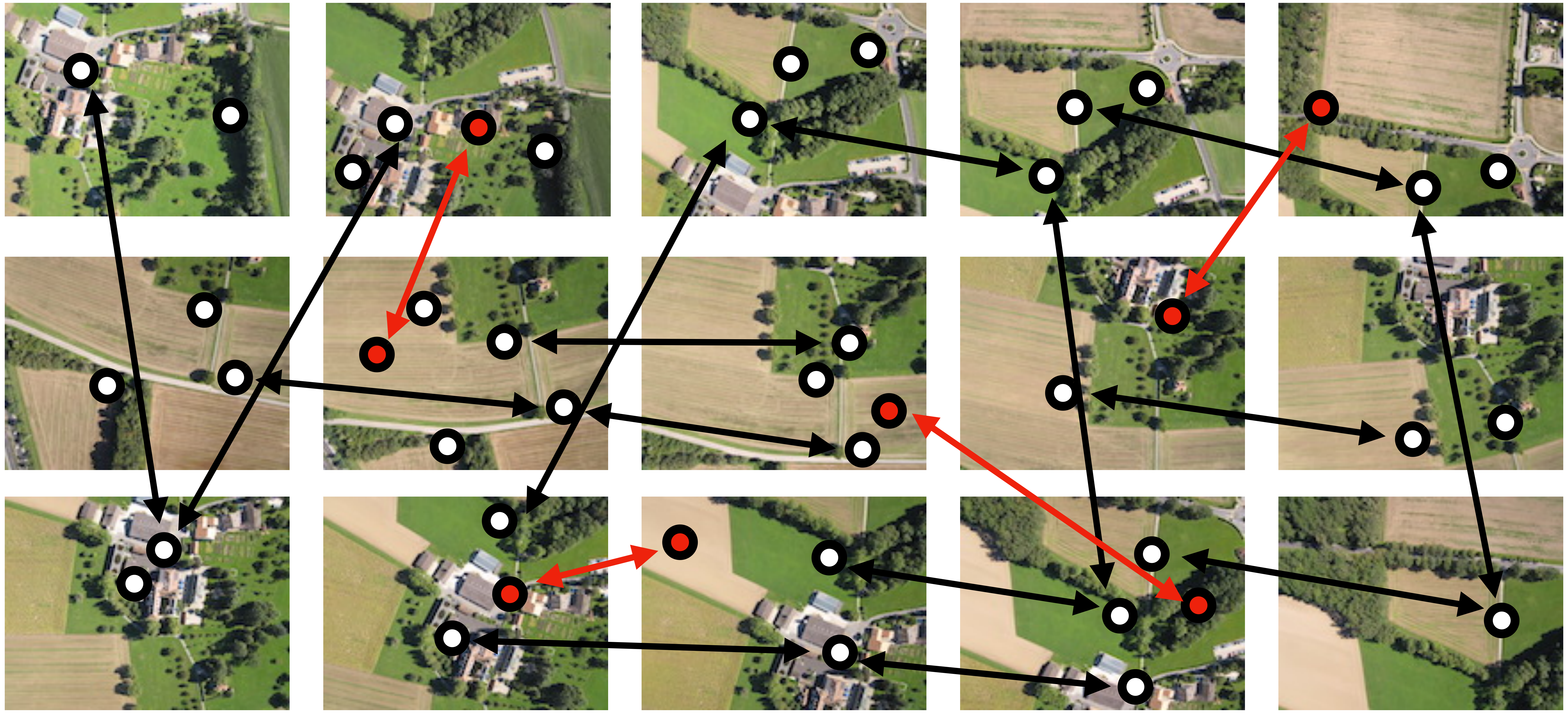
Align a set of images given a motion model (e.g., planar affine)



Step 1: Find all matches between images using SIFT

Multiview Image Alignment

Align a set of images given a motion model (e.g., planar affine)

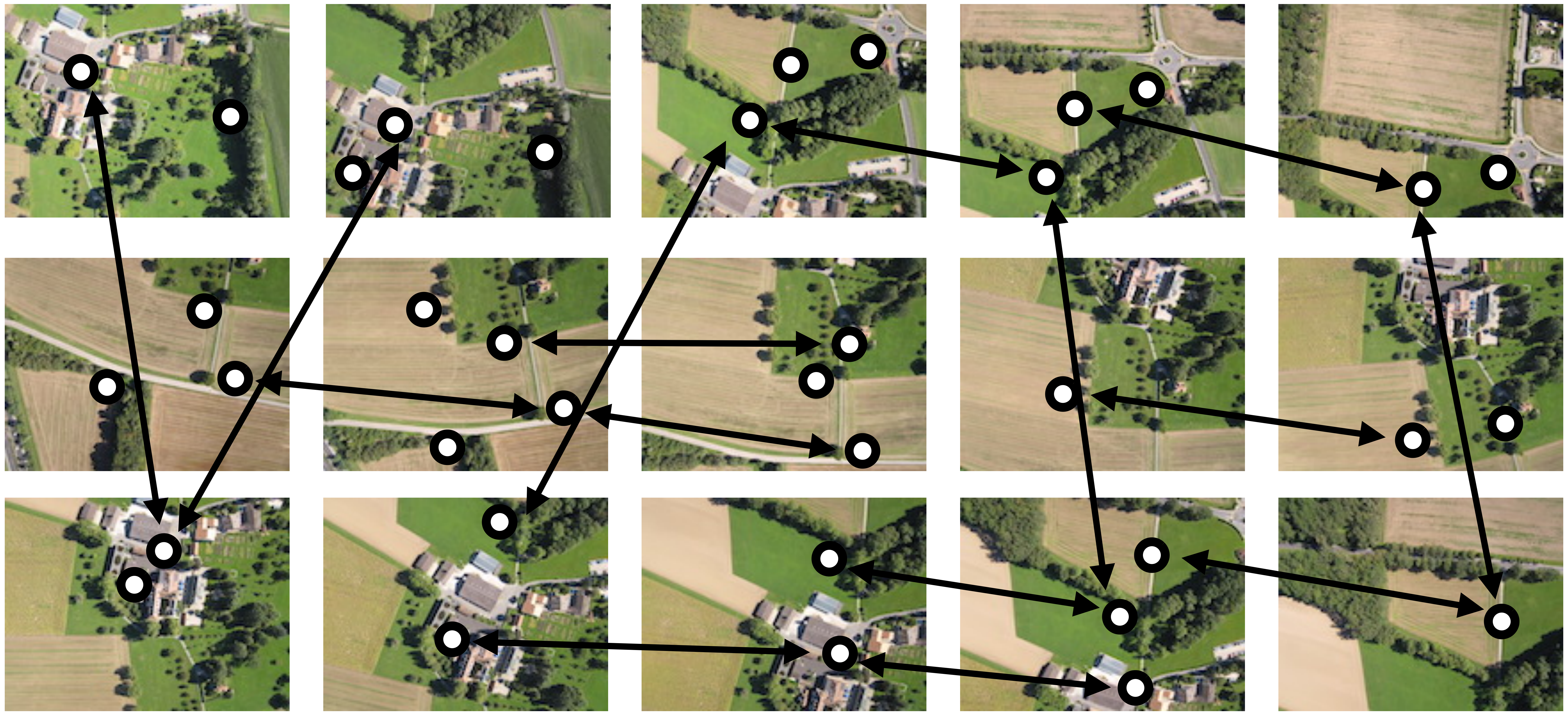


Step 1: Find all matches between images using SIFT

Step 2: Remove incorrect matches using RANSAC

Multiview Image Alignment

Align a set of images given a motion model (e.g., planar affine)



Step 1: Find all matches between images using SIFT

Step 2: Remove incorrect matches using RANSAC

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

4 outliers (blue, light blue, purple, pink)

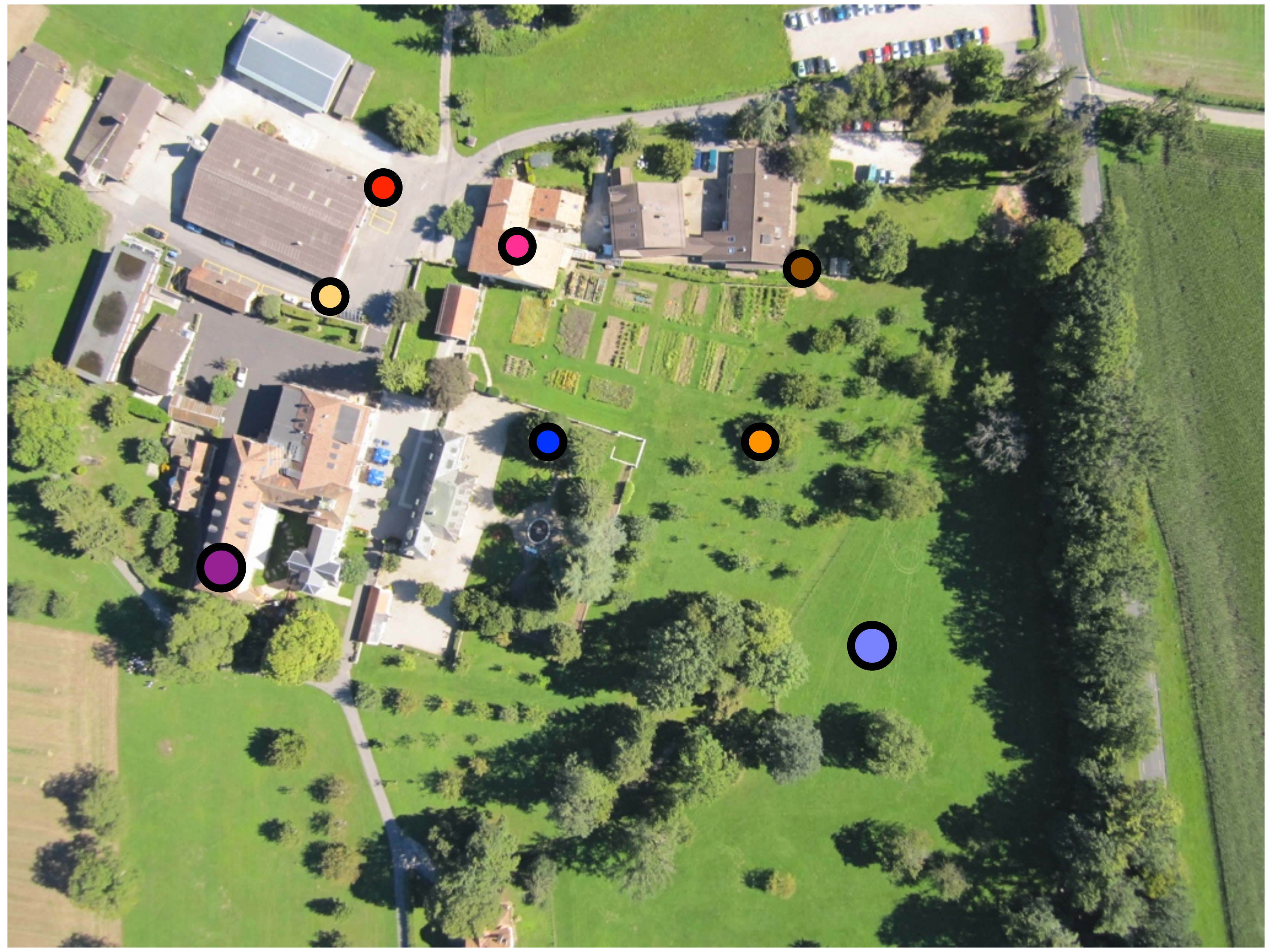
Recap: Image **Alignment + RANSAC**

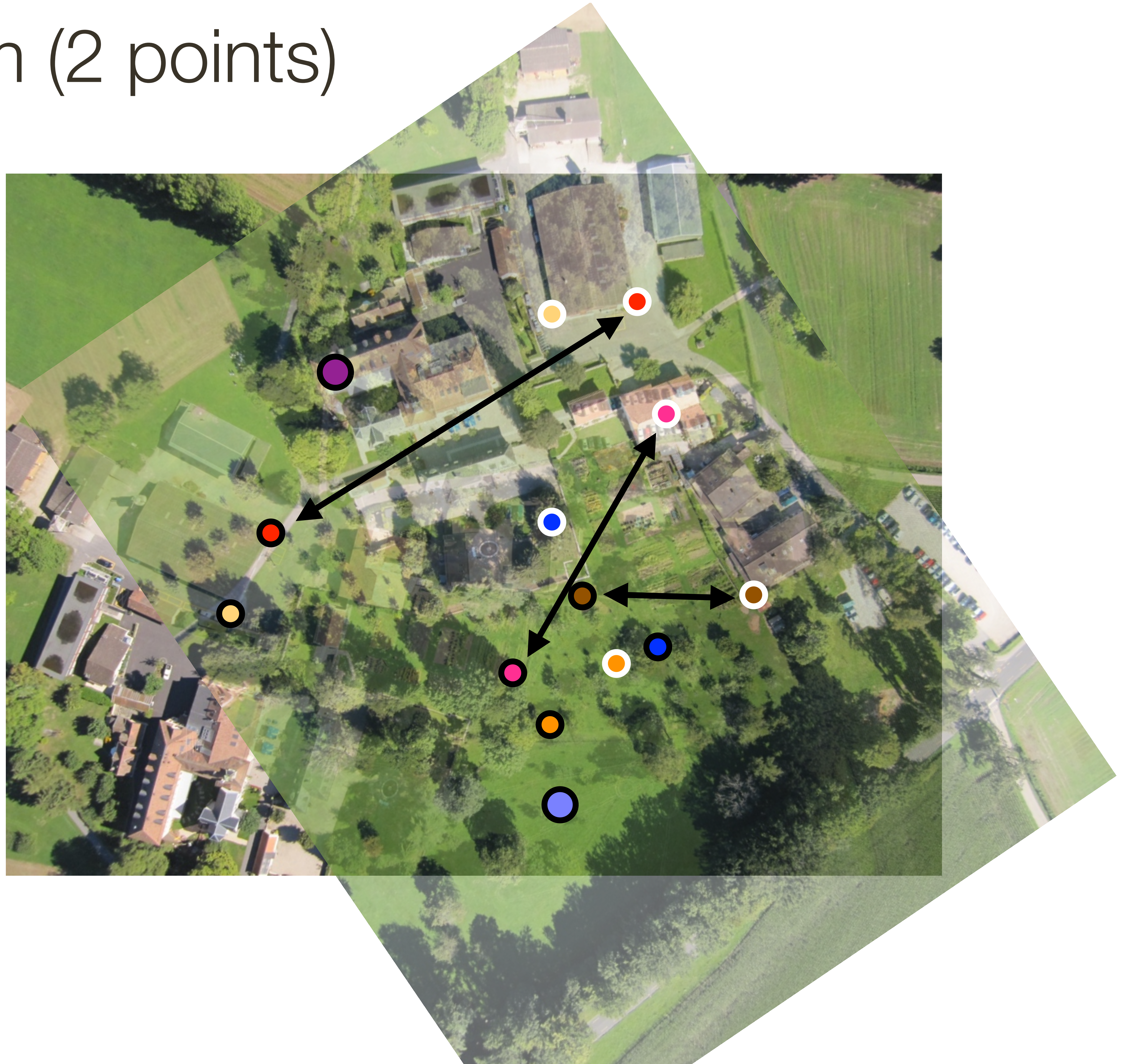
RANSAC solution for Similarity Transform (2 points)

4 inliers (red, yellow, orange, brown),
4 outliers (blue, light blue, purple, pink)

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)





check match distances

light blue, purple

#inliers = 2

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose pink, blue
warp image
check distances

#inliers = 2

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

Recap: Image **Alignment + RANSAC**

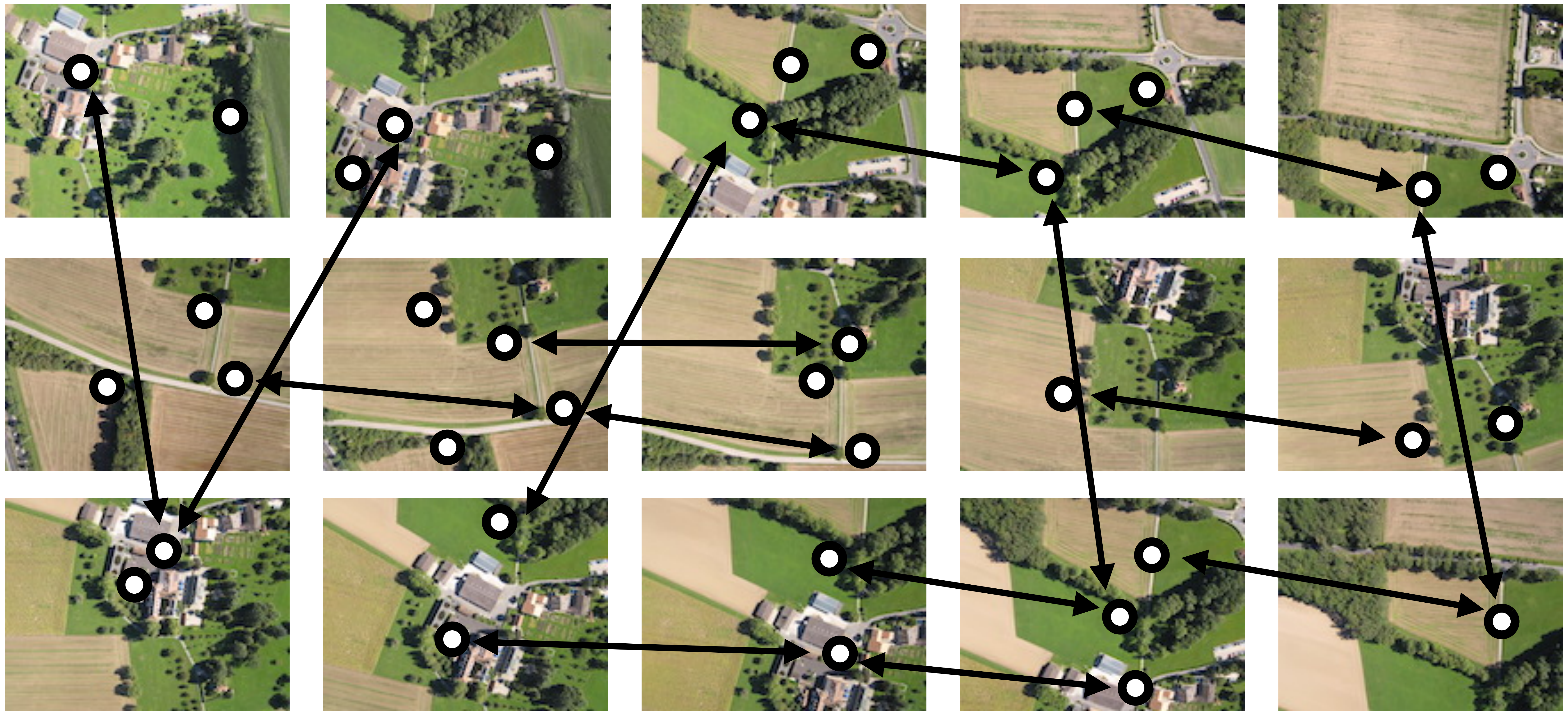
RANSAC solution for Similarity Transform (2 points)

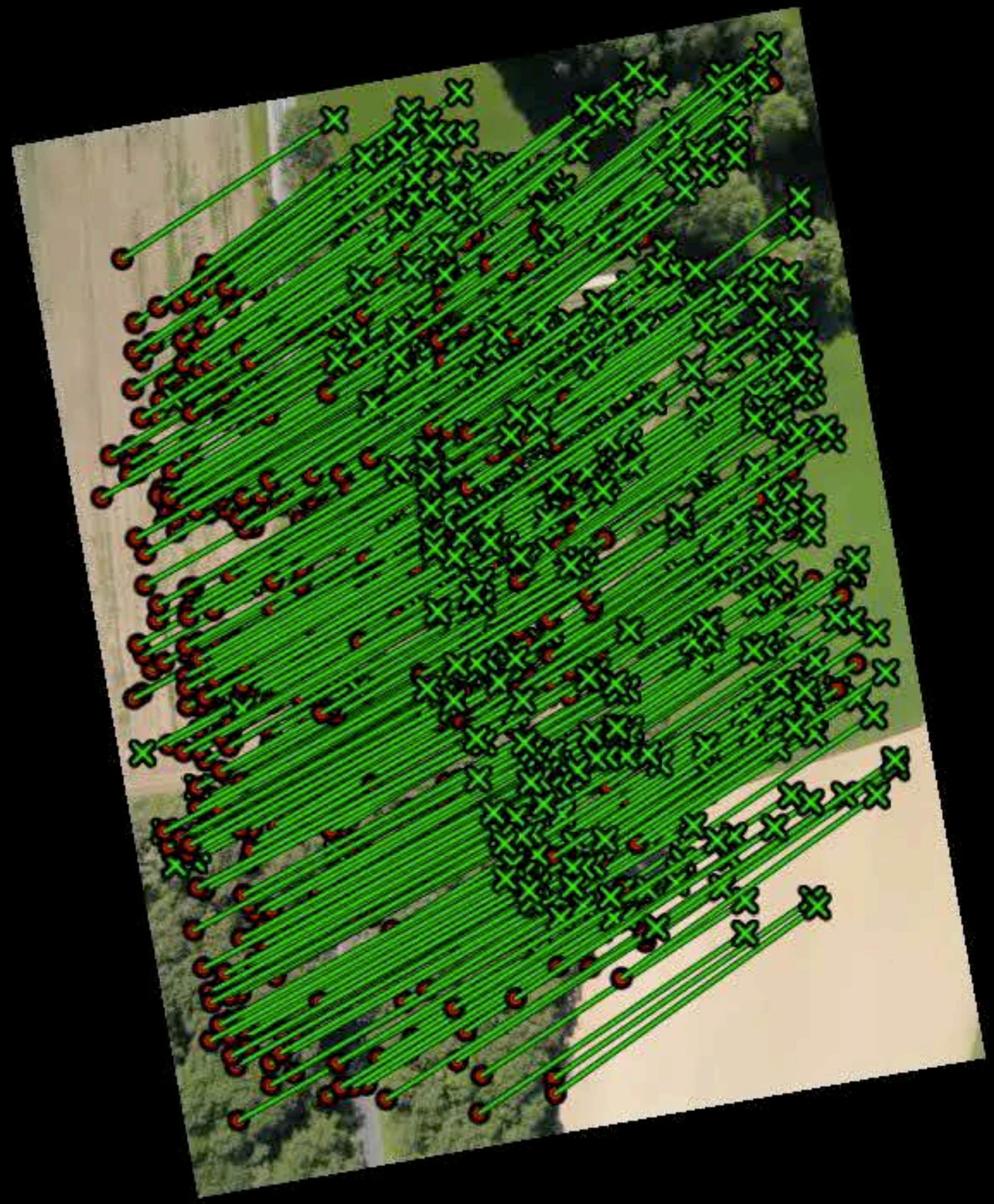
choose red, orange
warp image
compute distances

#inliers = 4

Planar Image Alignment

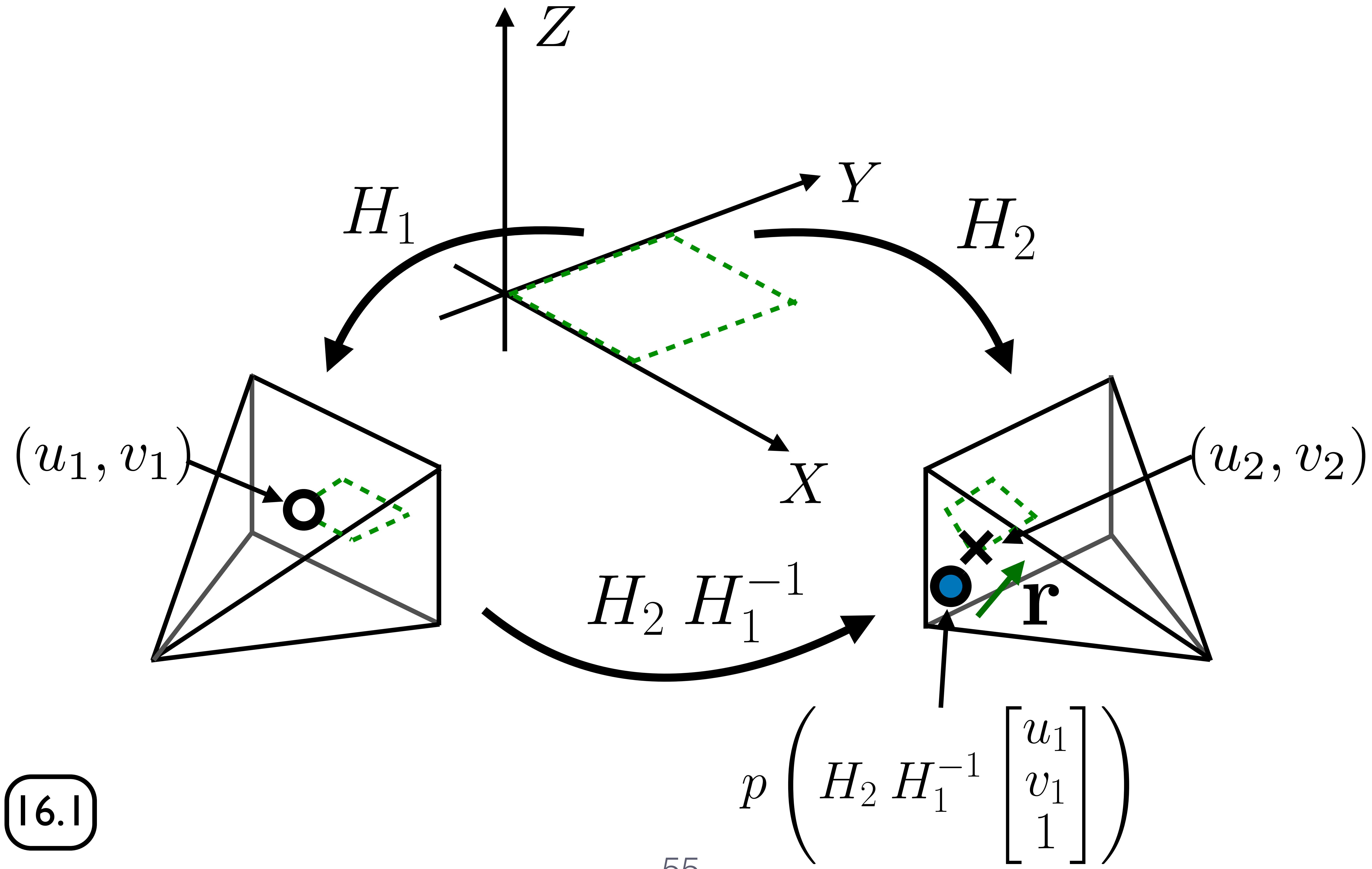
- Given a clean set of correspondences, align all images





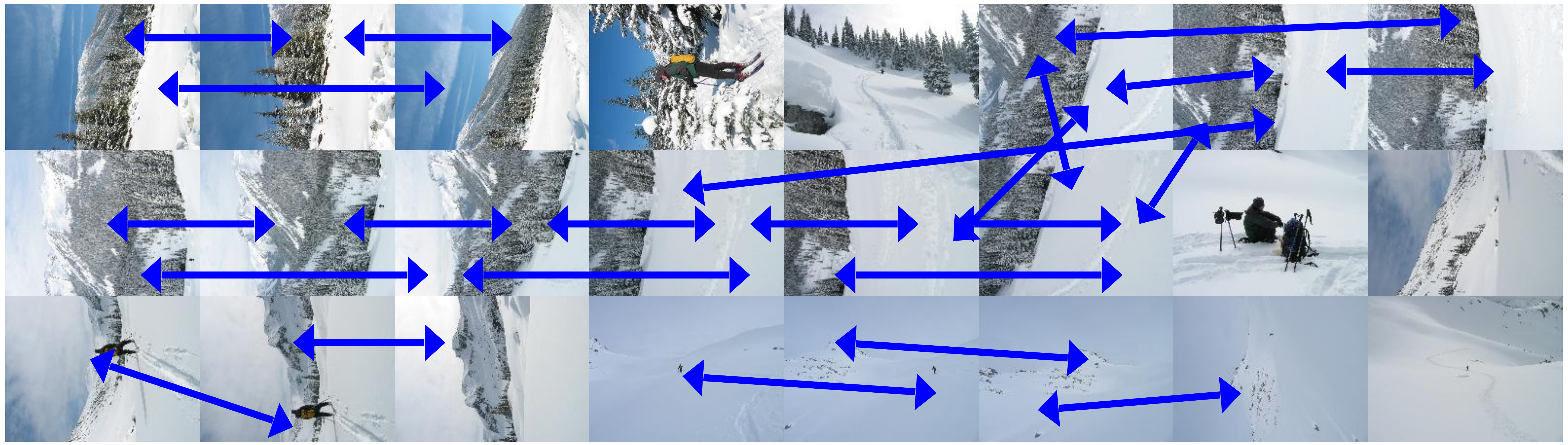
Multiview Image Alignment

Residual = vector between observed feature and projection

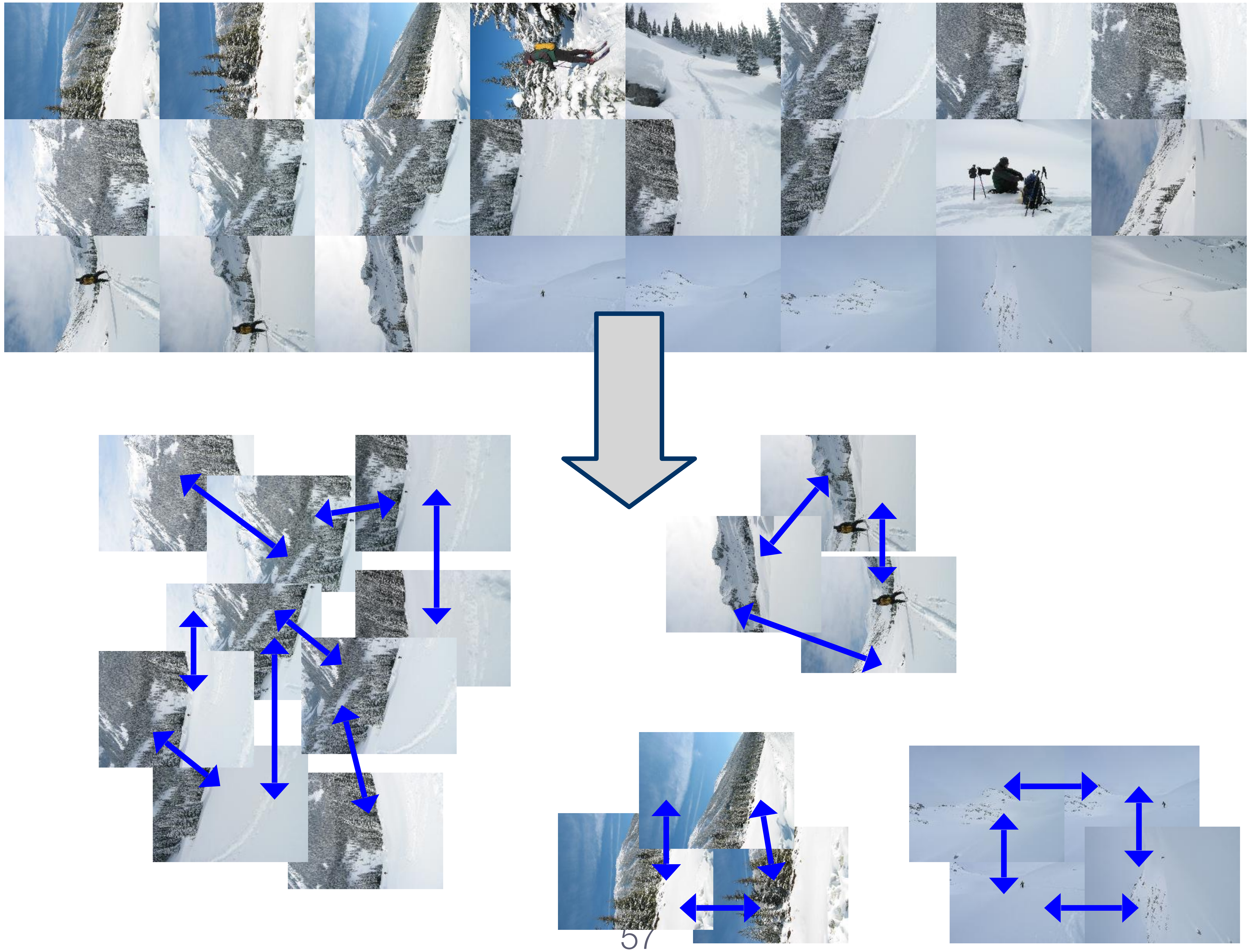


16.1

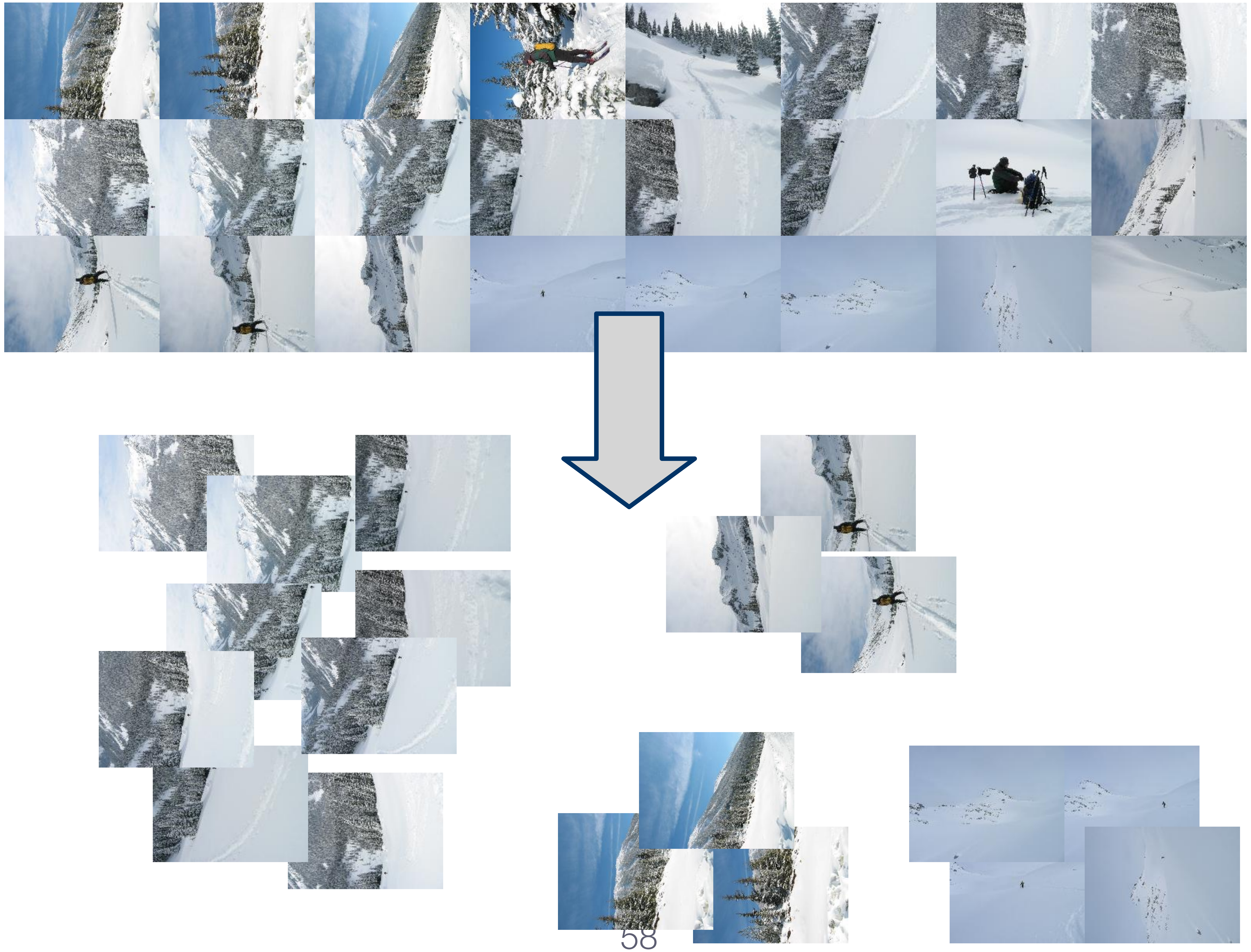
Panorama Recognition



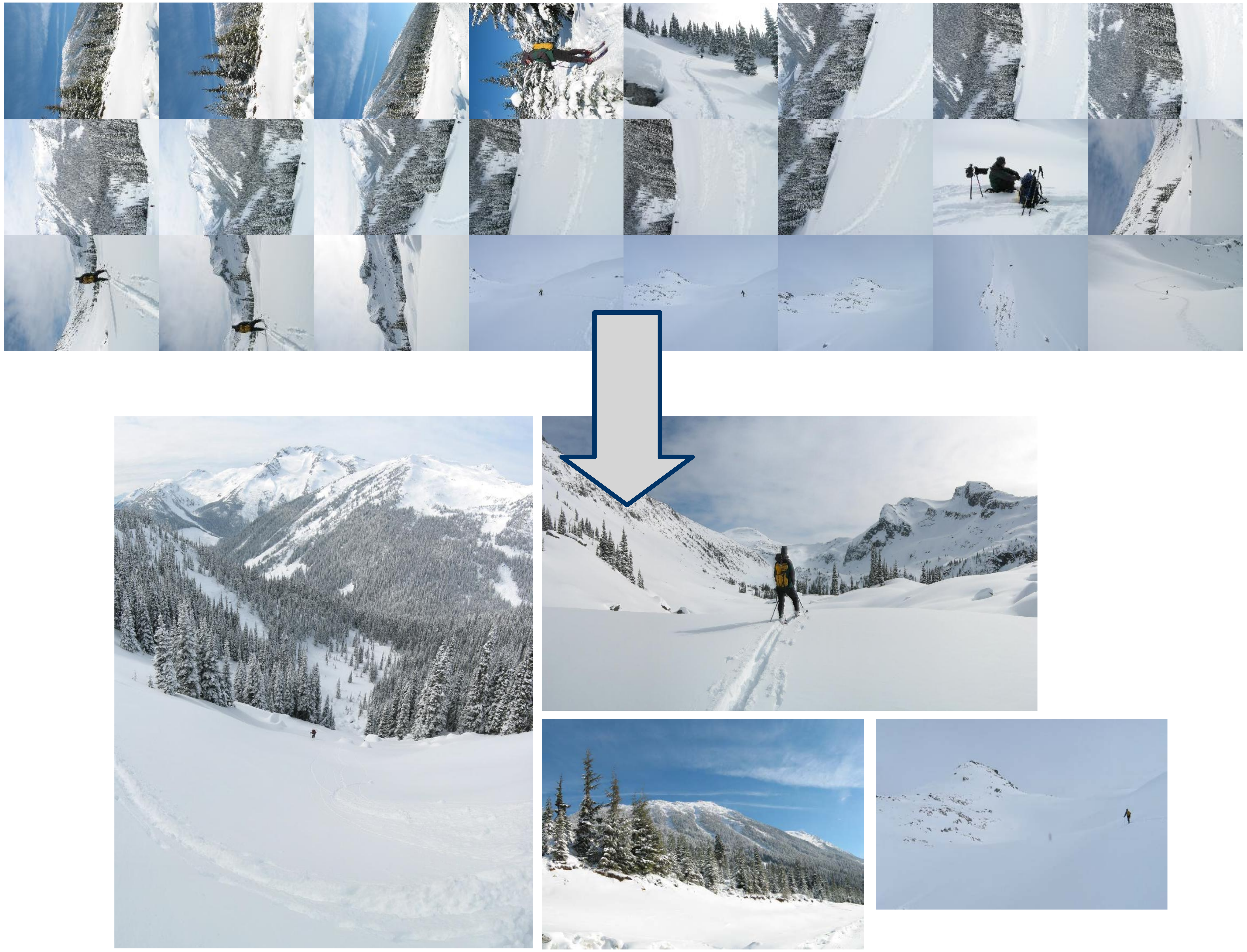
Panorama Recognition



Panorama Recognition



Panorama Recognition



Building a panorama

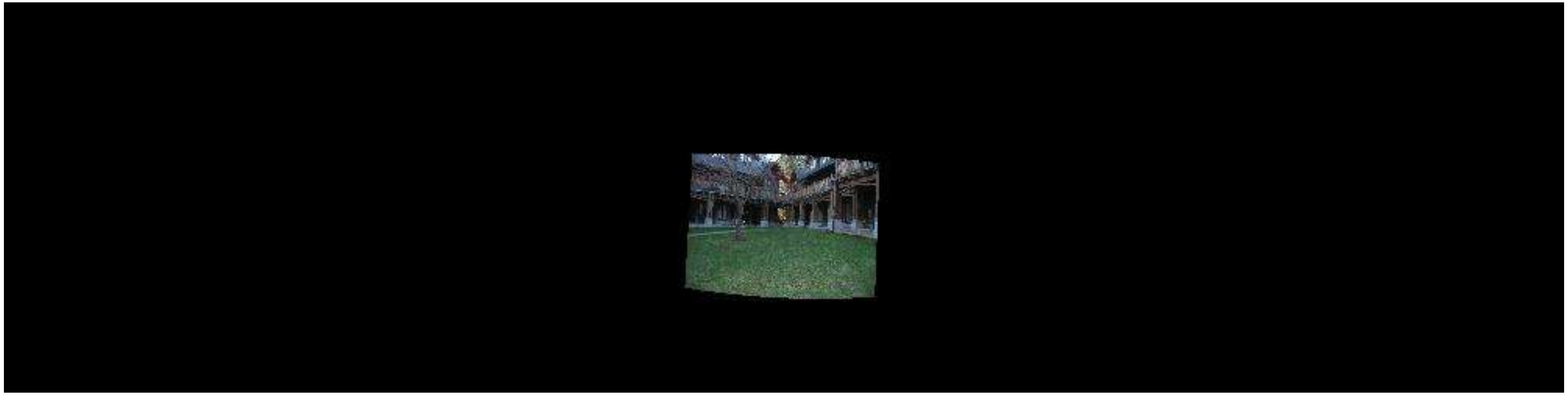


Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

Building a panorama

Figure Credit: Matthew Brown and David Lowe

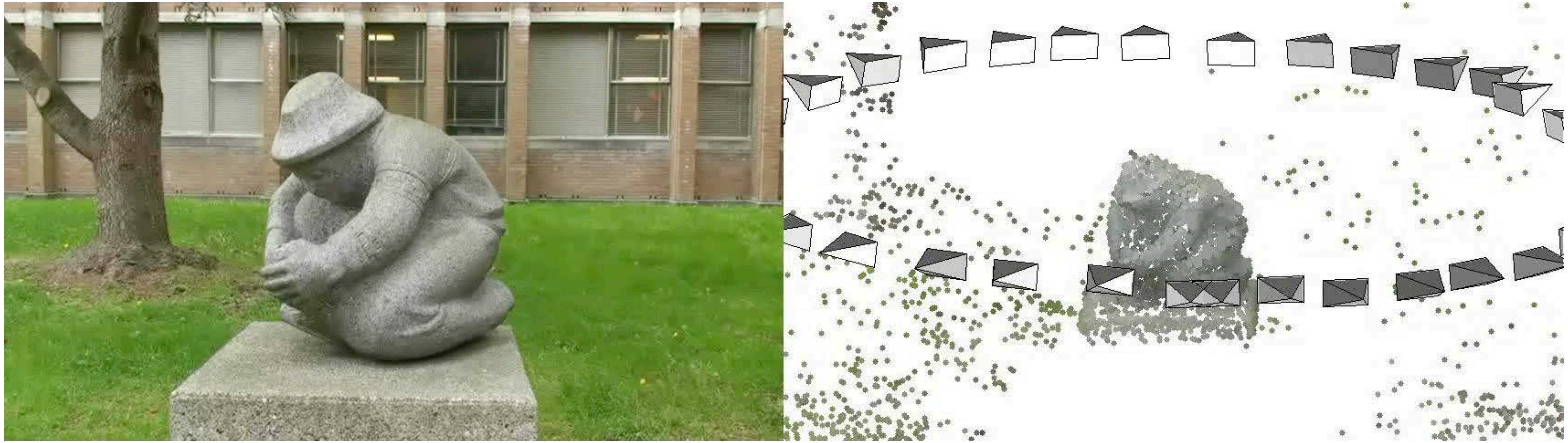
Panorama Stitching

- We can concatenate pairwise homographies, but over time multiple pairwise mappings accumulate errors
- We use global alignment (bundle adjustment) to close the gap

Structure from Motion

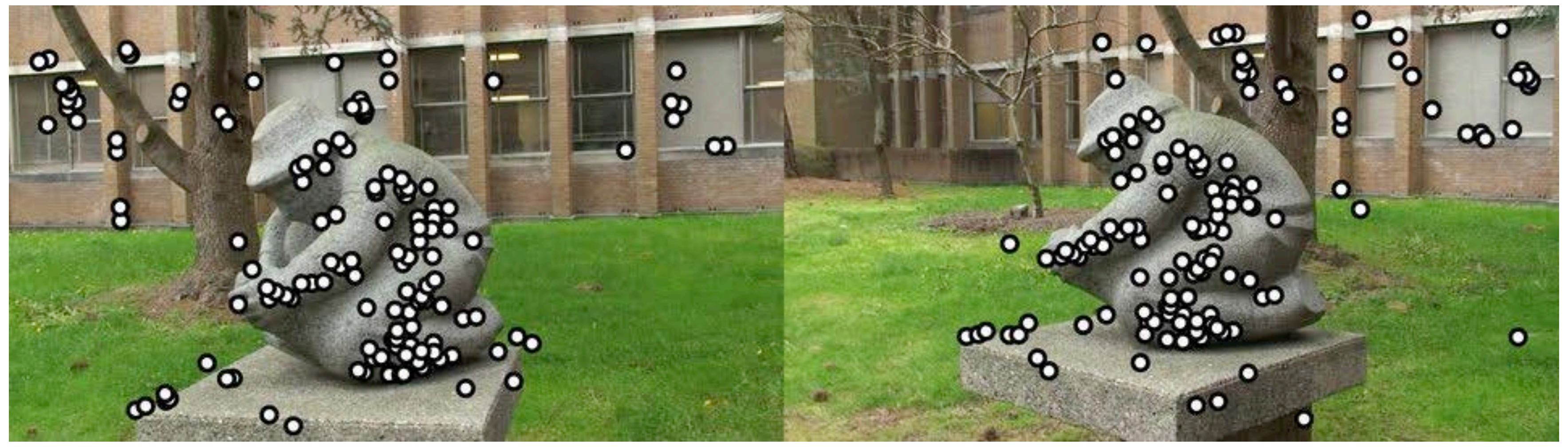
Given an (unordered) set of input images, compute cameras and 3D structure of the scene

Structure from Motion



2-view Structure from Motion

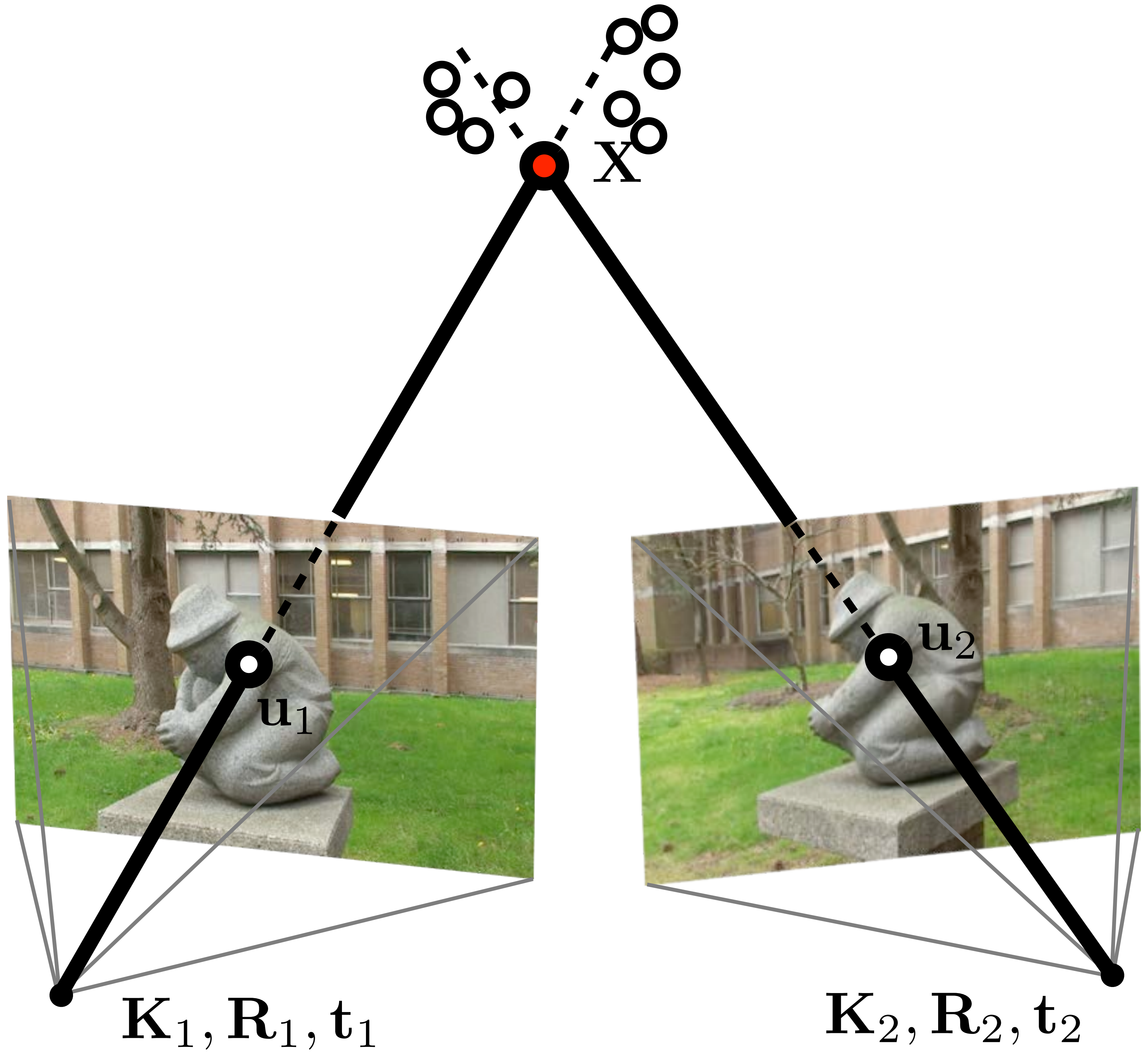
- We can use the combination of SIFT/RANSAC and triangulation to compute 3D structure from 2 views



Raw SIFT matches

RANSAC for Epipolar Geom

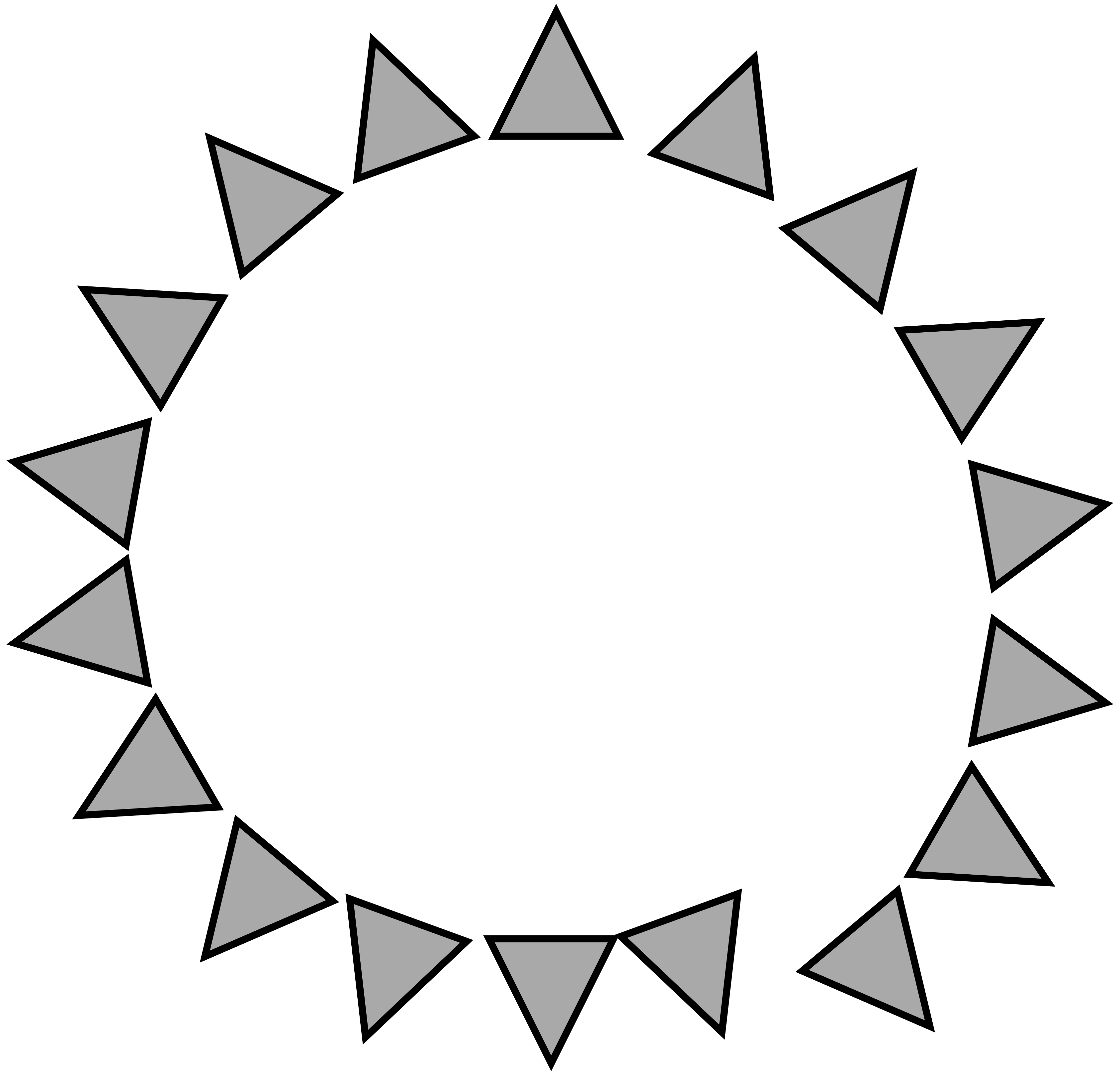
Extract R, t



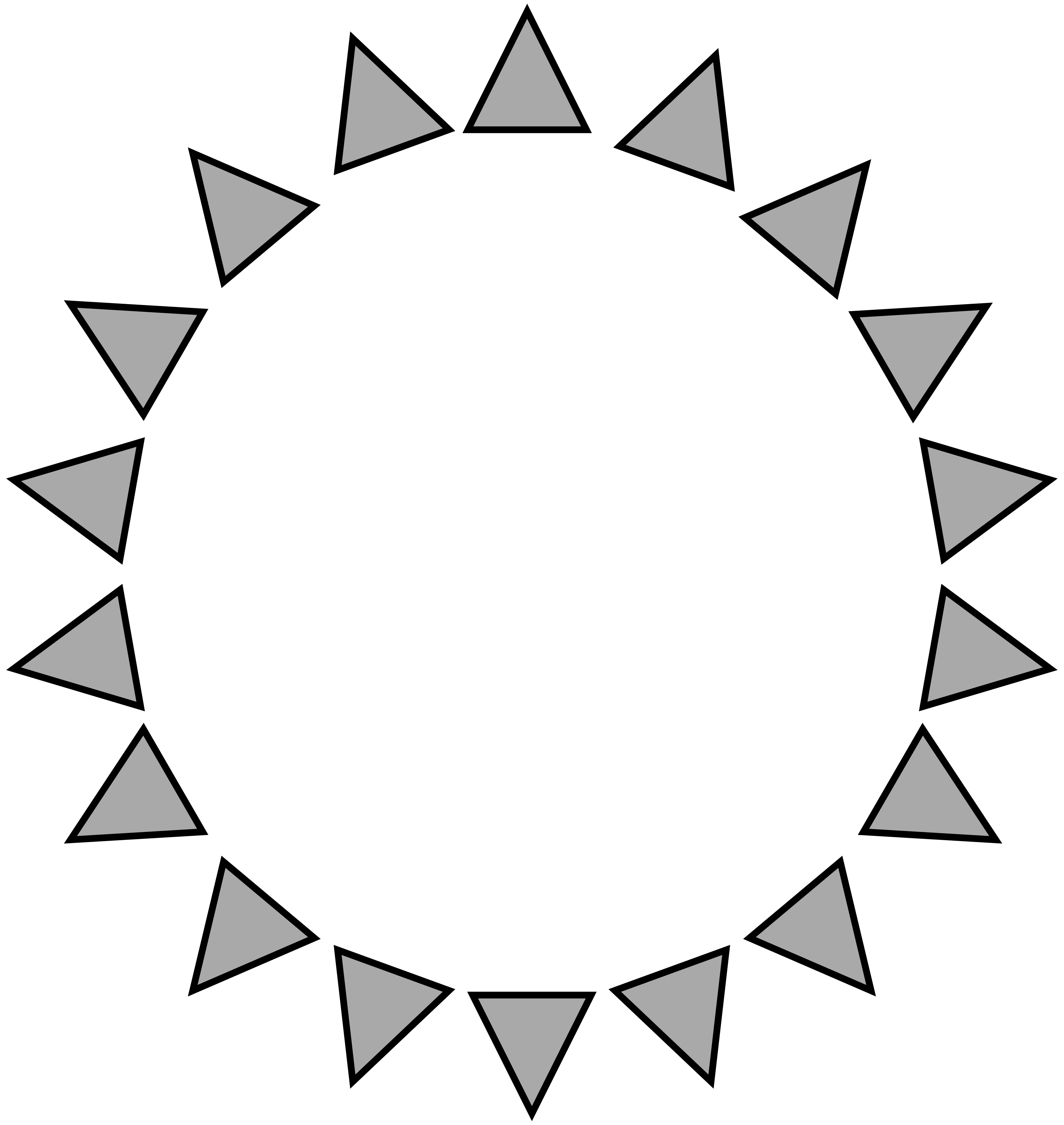
Triangulate to 3D Point Cloud

Global Alignment

- Concatenation of pairwise R, t estimates results in drift, e.g.,



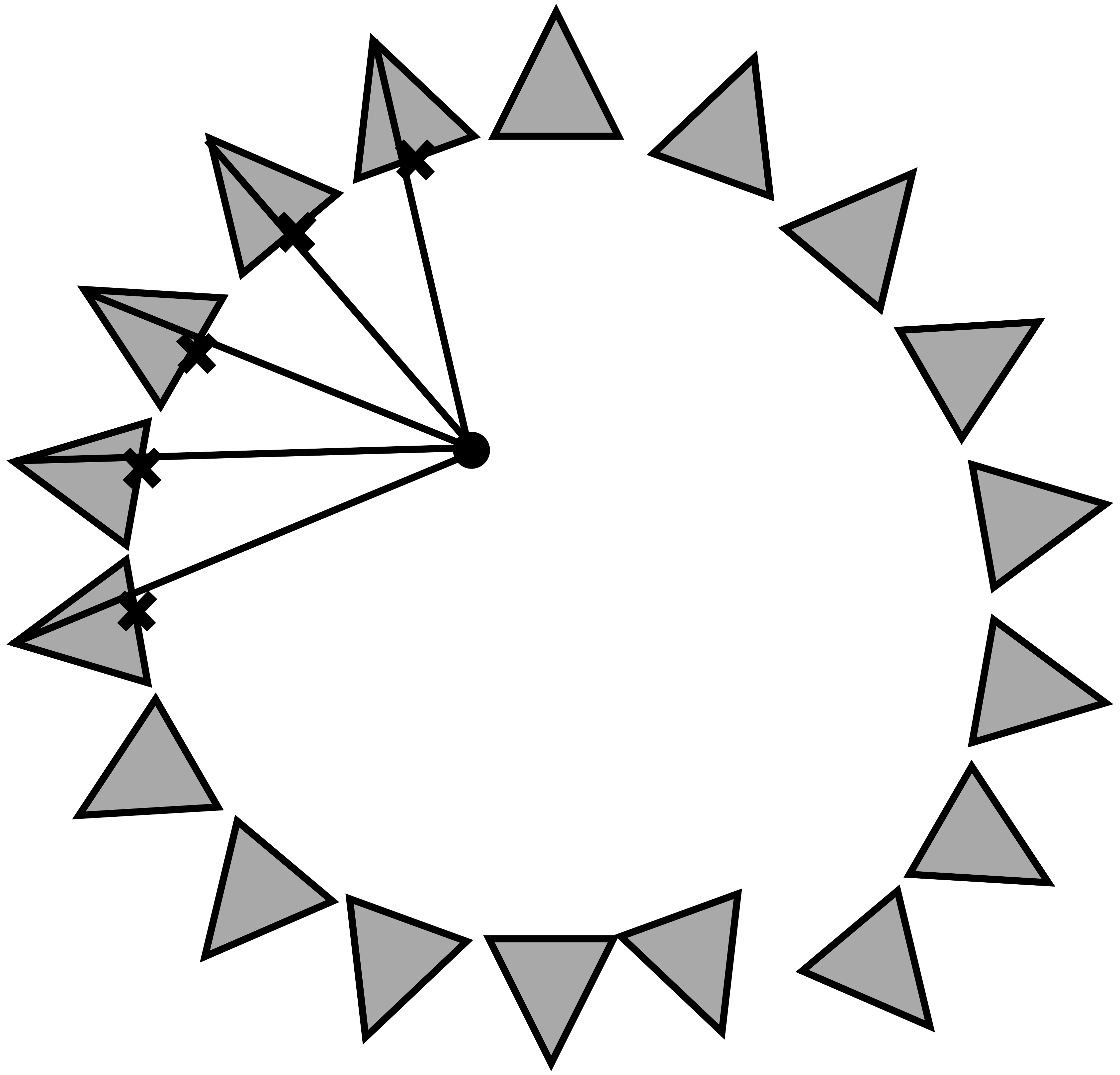
Pairwise alignment



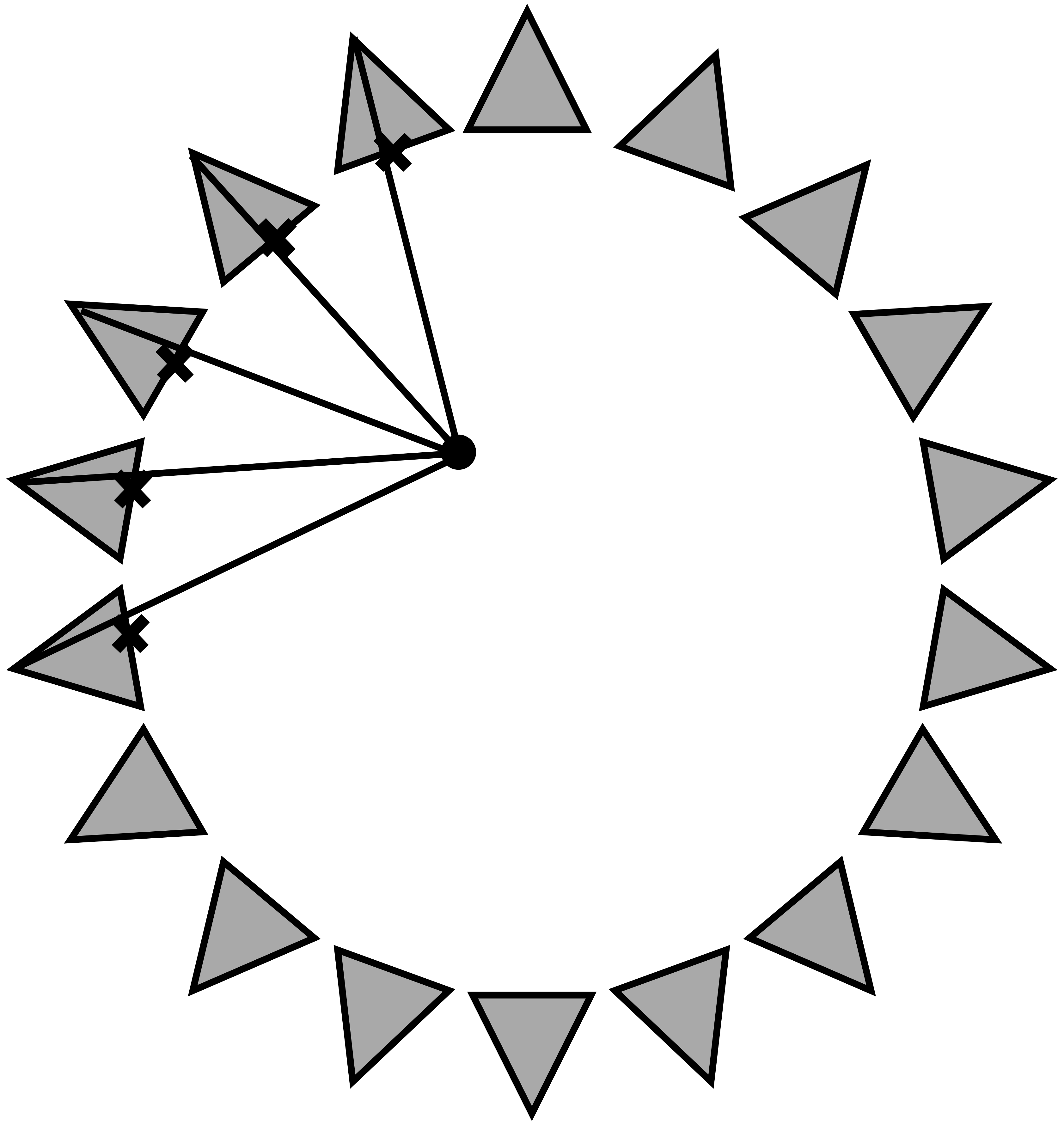
Global alignment

Global Alignment

- Concatenation of pairwise R, t estimates results in drift, e.g.,



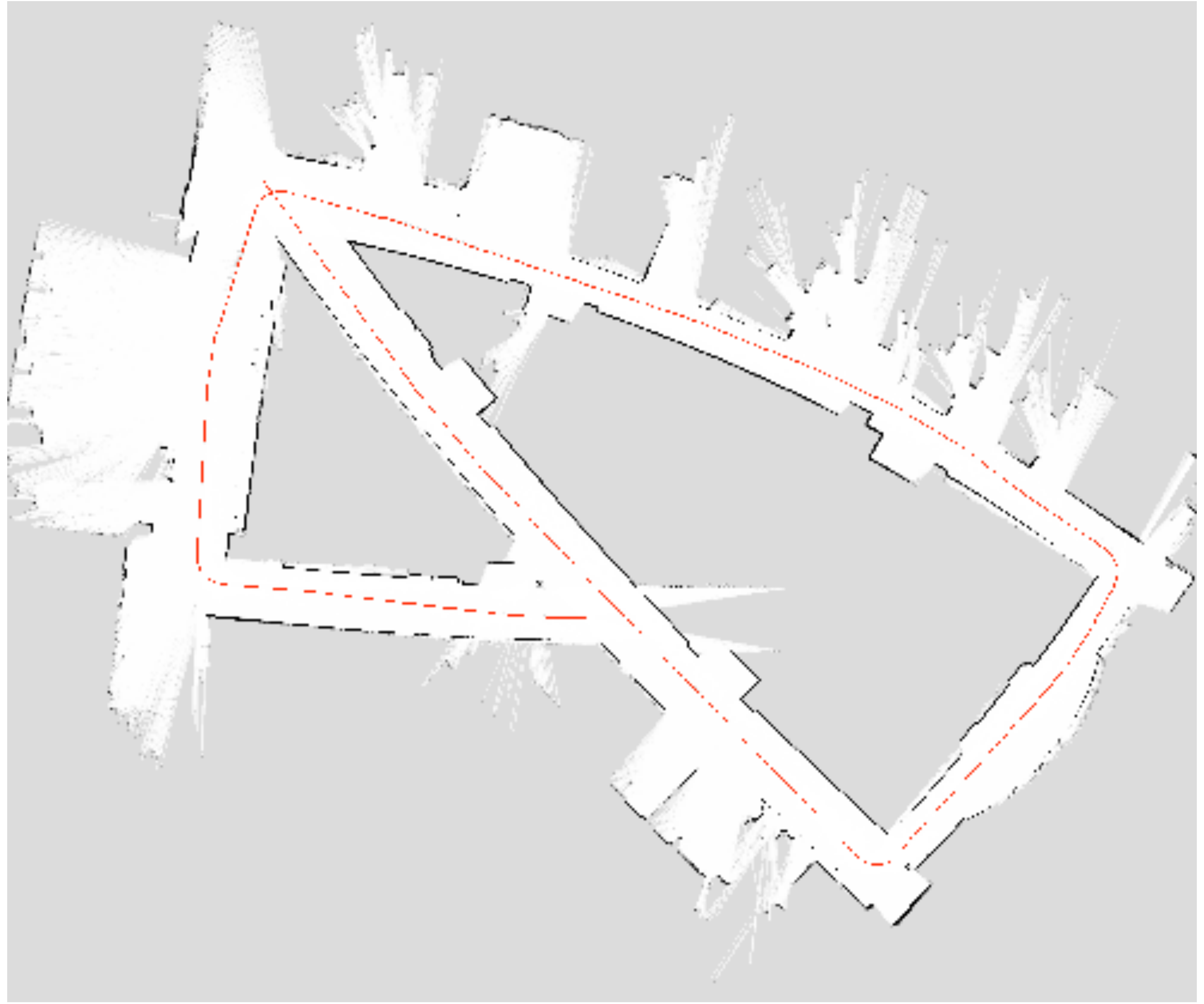
Pairwise alignment



Global alignment

Global Alignment

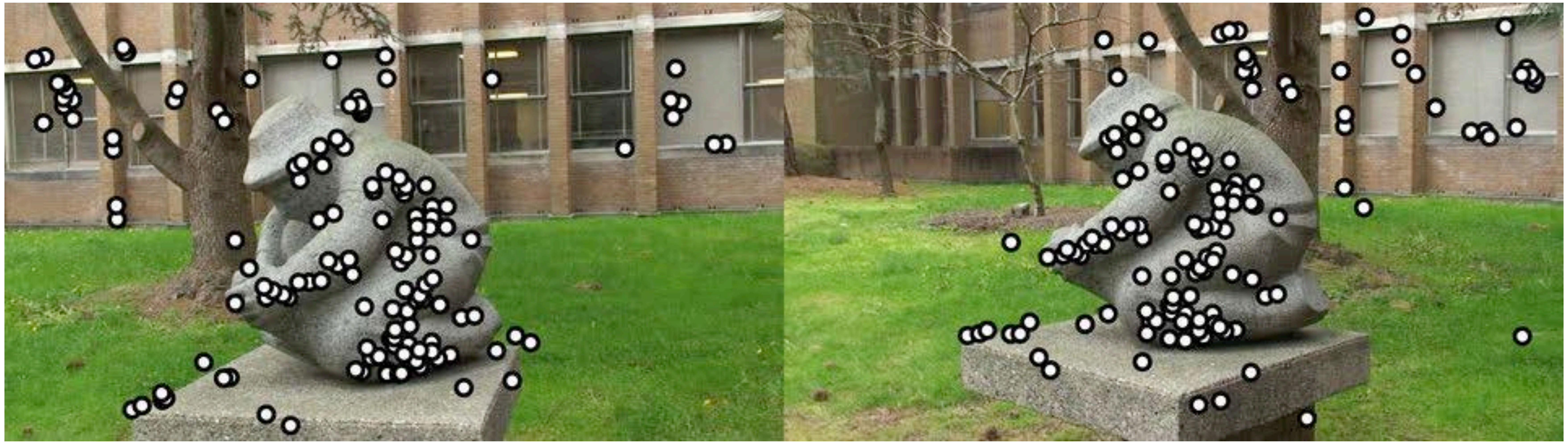
- In robotic navigation frame-frame alignment also causes drift



We can use **bundle adjustment** to close the gap

[Kaess Dellaert 2010]

RANSAC for 3D Matches

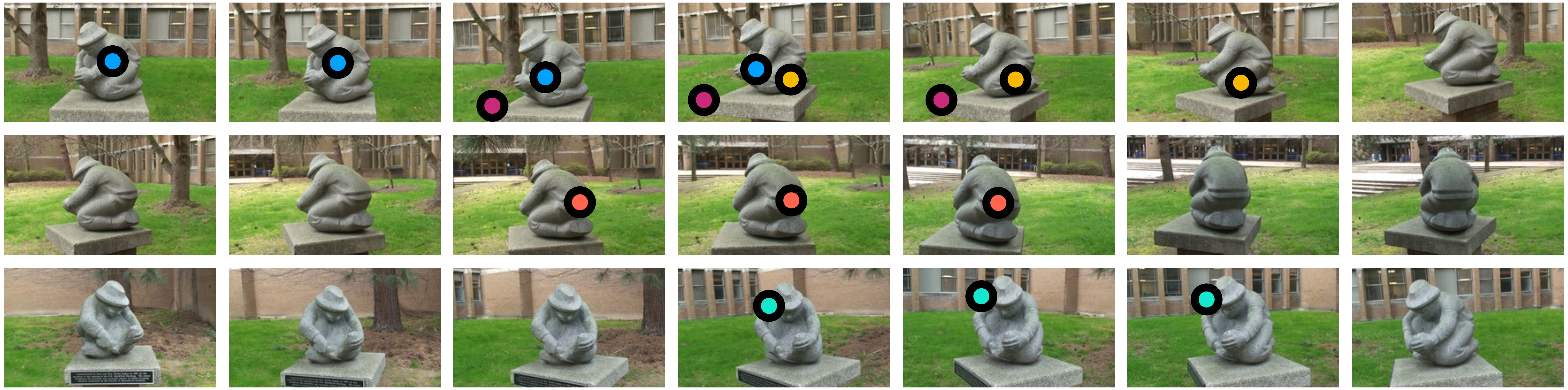


Raw feature matches (after ratio test filtering)

Solved for RANSAC inliers

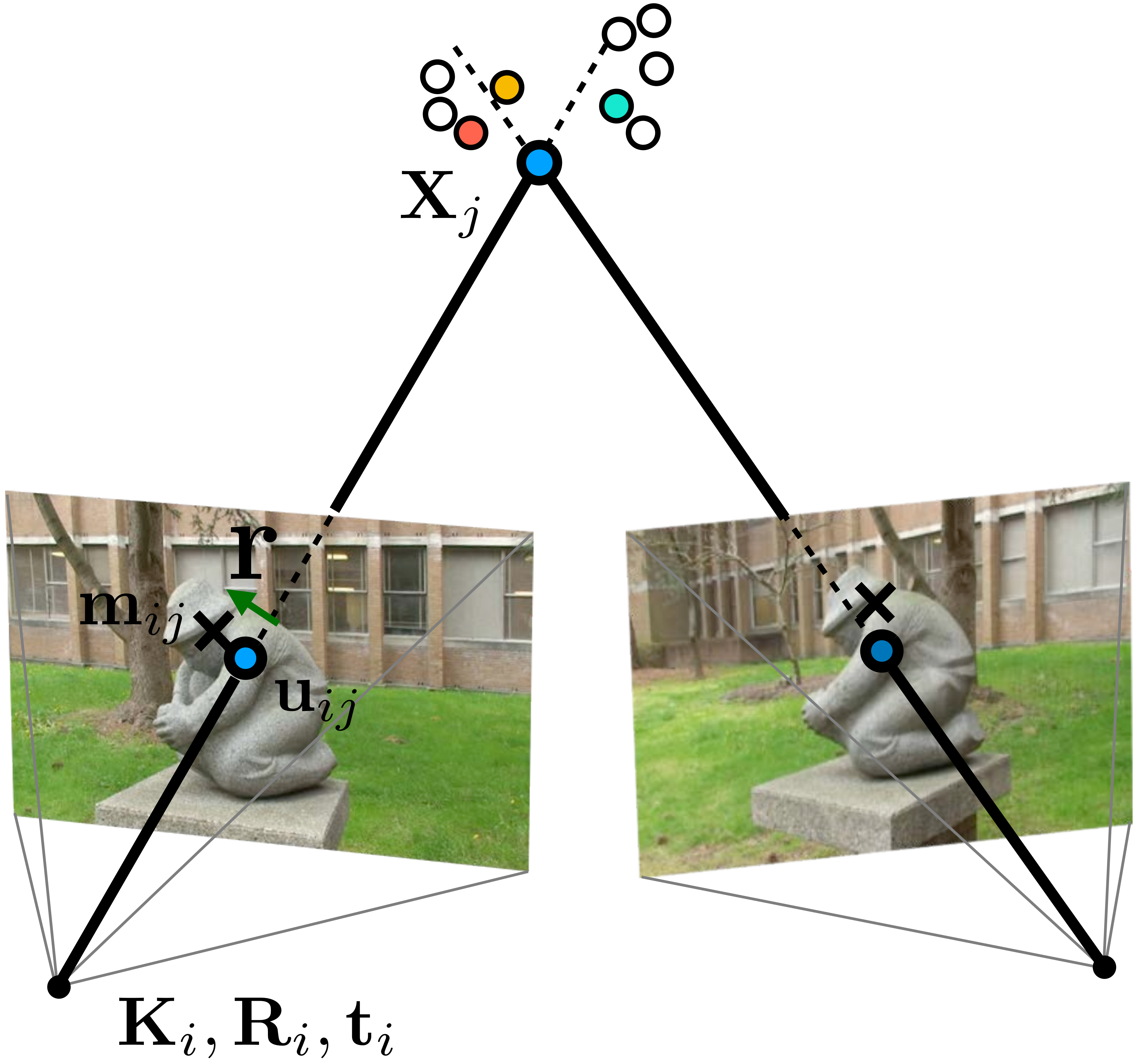
Feature Tracking

- Form feature tracks by combining pairwise feature matches



- Tracked features become individual 3D points in the reconstruction
- Features matched across 3 or more views provide strong constraints on the 3D reconstruction

Bundle Adjustment



- Minimise errors projecting 3D points into all images

$$e = \sum_{i \in \text{images}} \sum_{j \in \text{points}} |\mathbf{r}_{ij}(\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)|^2$$

[Szeliski 11.4]

Bundle Adjustment

- Full bundle adjustment (optimise all cameras and points):

$$e = \sum_{i \,\epsilon\, \text{images}} \sum_{j \,\epsilon\, \text{points}} |\mathbf{r}_{ij}(\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)|^2$$

- Triangulation (optimise points, fixed cameras):

$$e = \sum_{i \,\epsilon\, \text{images}} \sum_{j \,\epsilon\, \text{points}} |\mathbf{r}_{ij}(\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)|^2$$

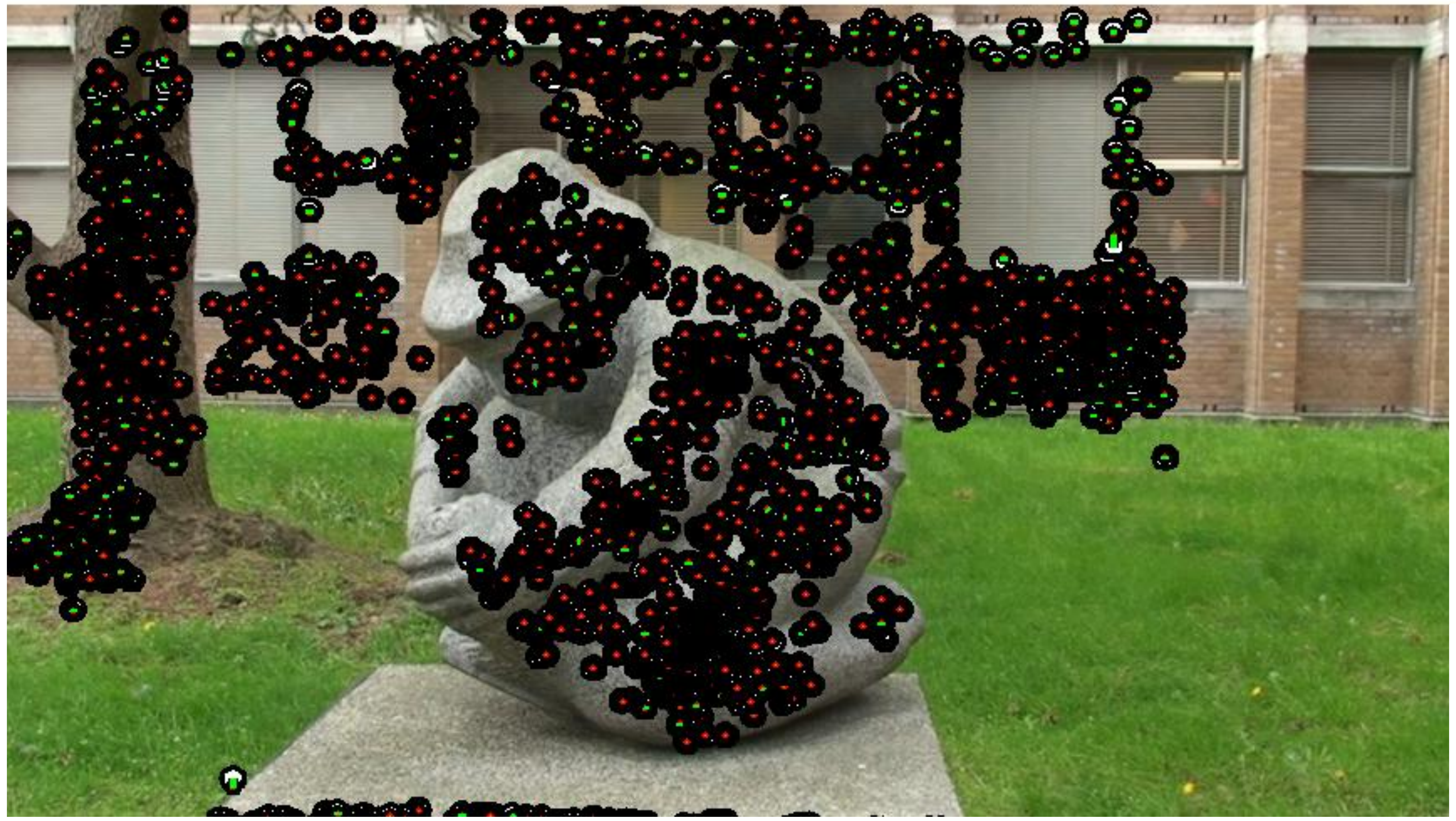
- Pose estimation for camera i:

$$e = \sum_{j \,\epsilon\, \text{points}} |\mathbf{r}_{ij}(\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)|^2$$

(optimised parameters are shown in red)

Bundle Adjustment

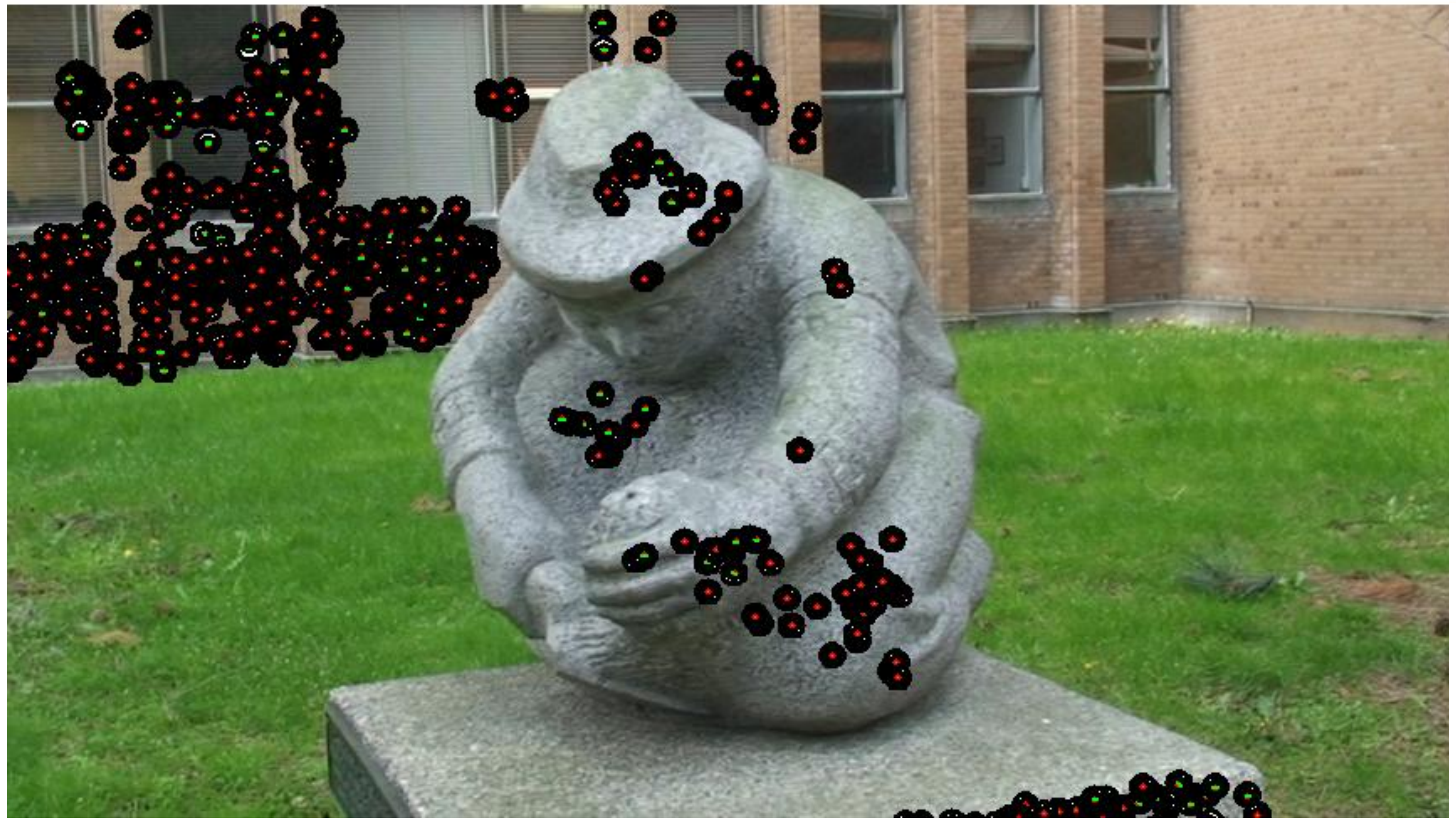
- Initialization with 3 views



Joint optimization of cameras and structure

Bundle Adjustment

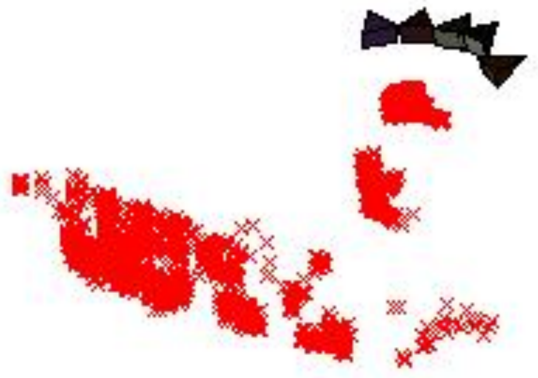
- Add camera 4



Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

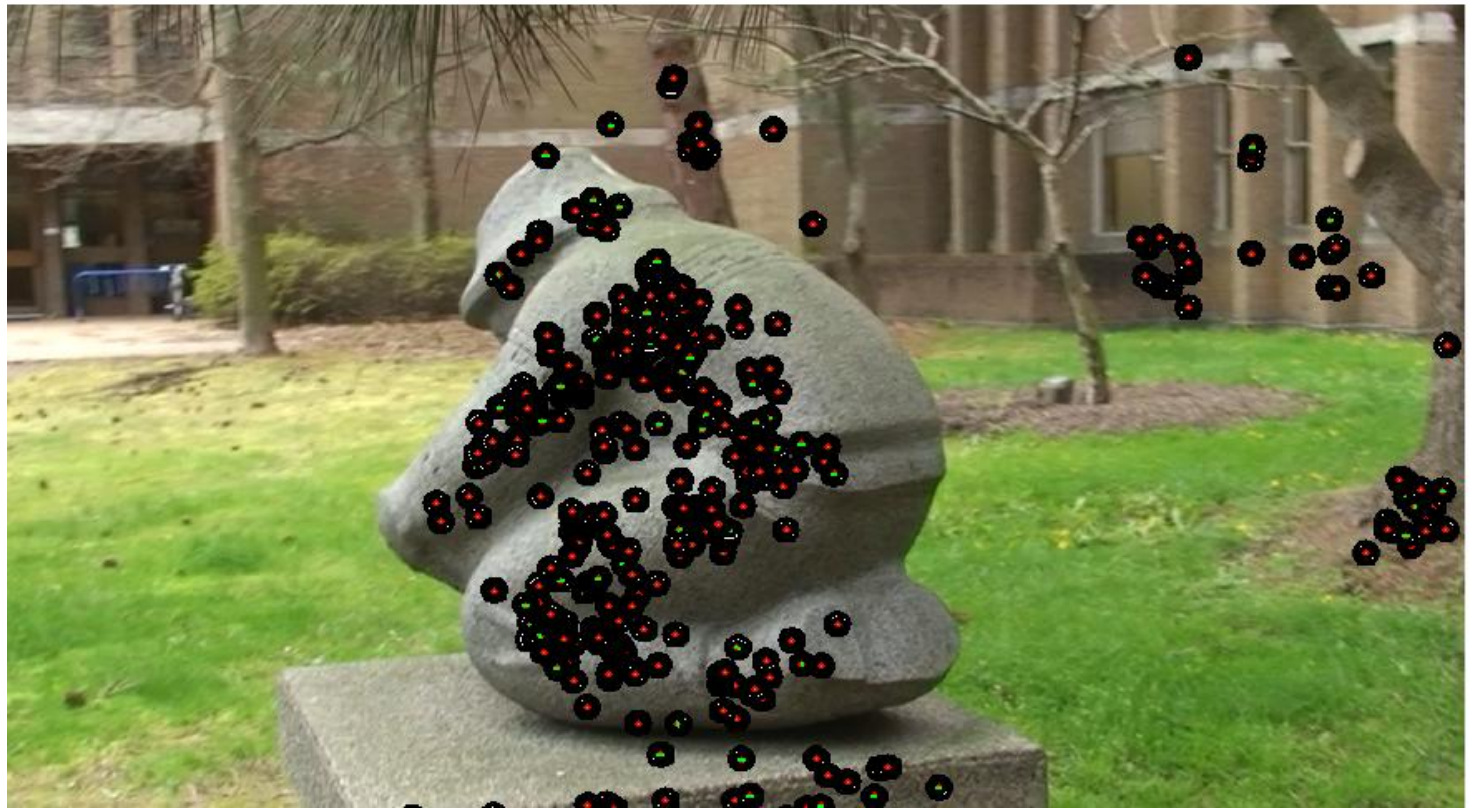
- Add camera 5



Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

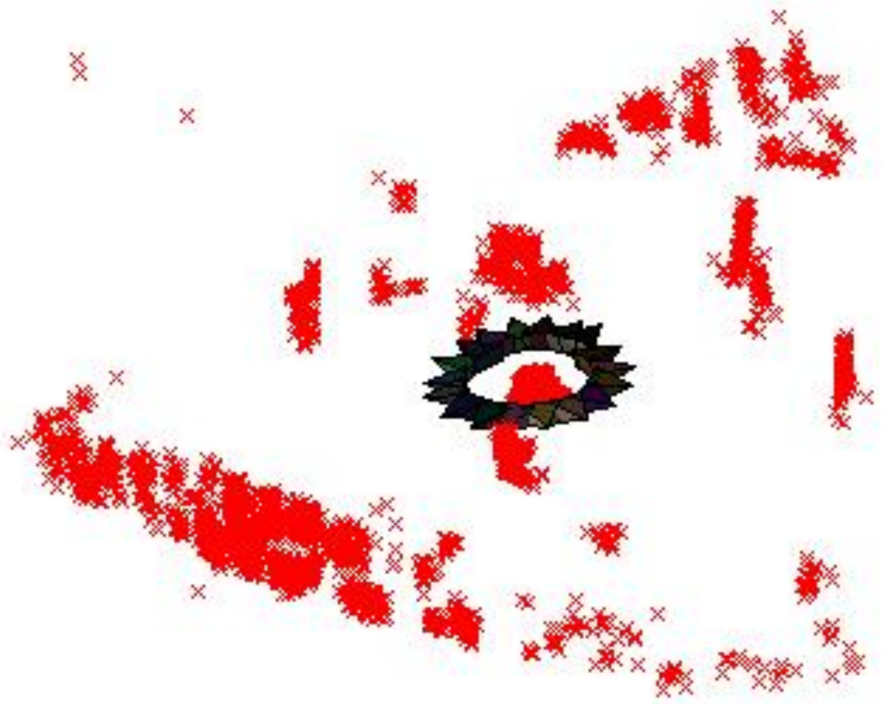
- Add camera 6



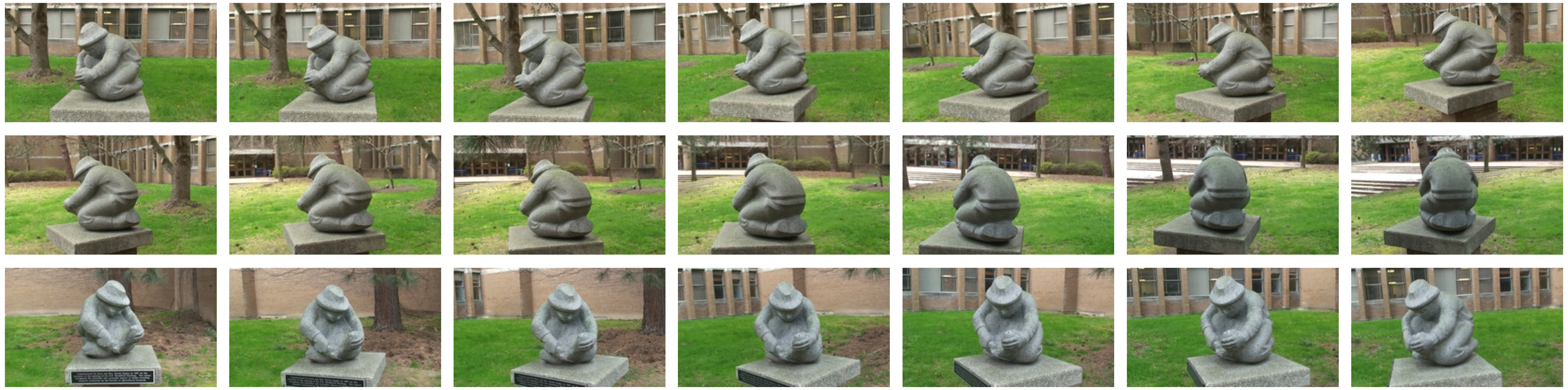
Estimate camera pose, add new 3D points, jointly optimize

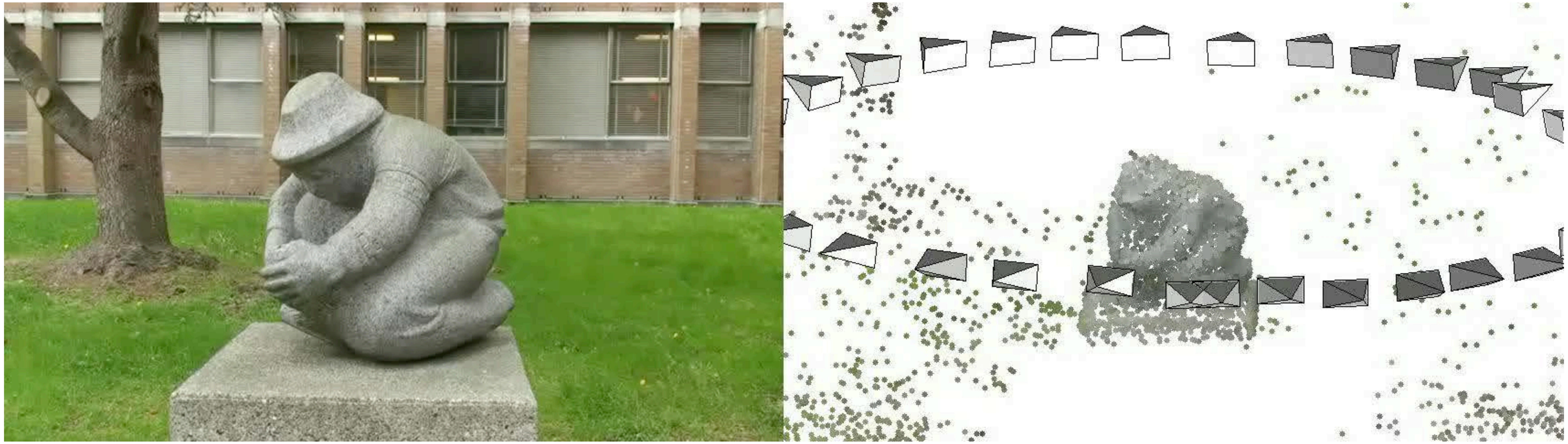
Bundle Adjustment

- Add remaining cameras in same way



Structure from Motion

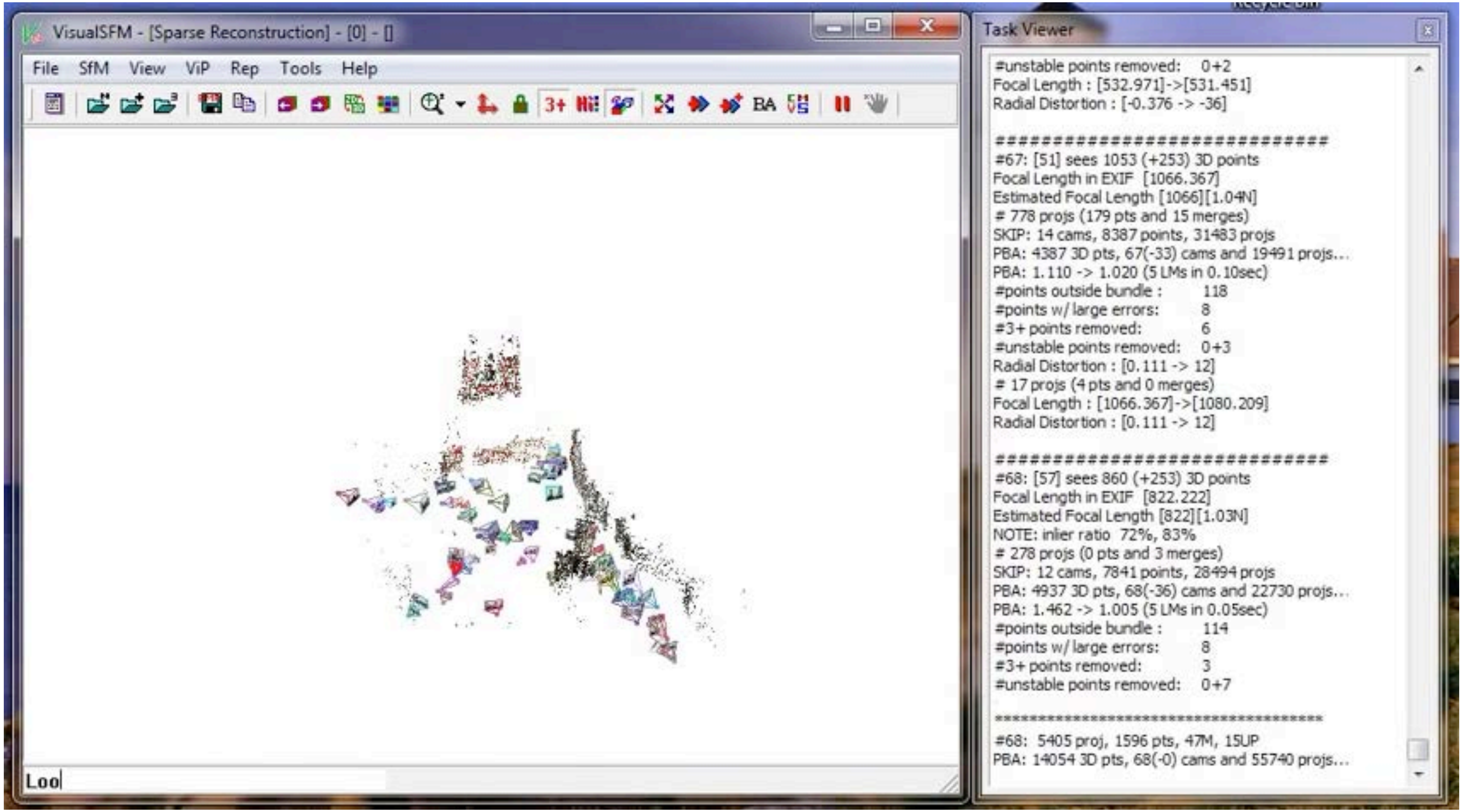




SFM recap

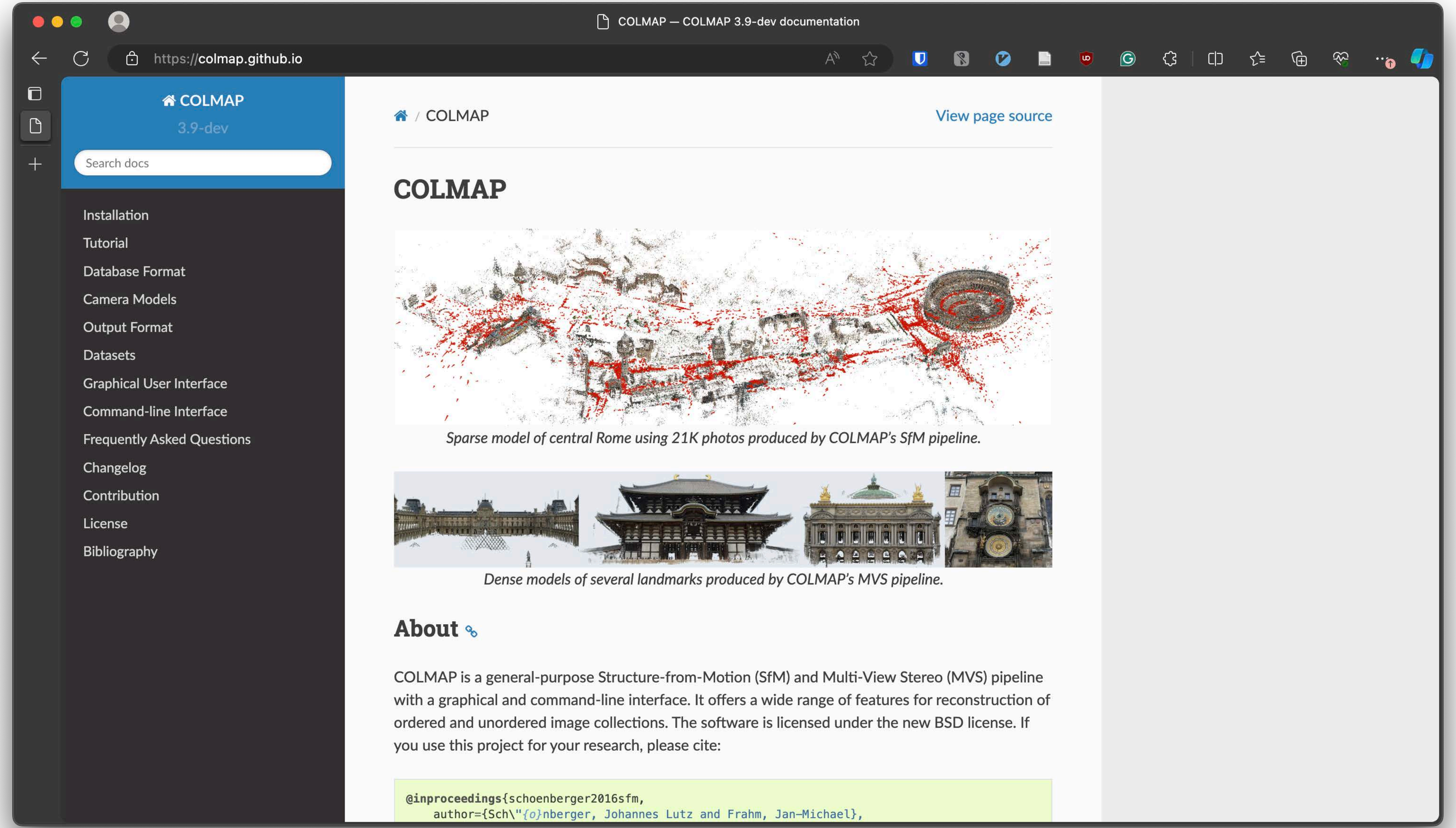
- Match features, e.g., SIFT, between all views
- Use RANSAC to reject outliers and estimate Epipolar Geometry / Camera matrices
- Form feature tracks by linking multiview matches
- Select an initialization set, e.g., 3 images with lots of matches and good baseline (parallax)
- Jointly optimize cameras R, t and structure X for this set
- Repeat for each camera:
  - Estimate pose R, t by minimising projection errors with existing X
  - Add 3D points corresponding to the new view and optimize
  - Bundle adjust optimizing over all cameras and structure

Visual SFM



[ccwu.me/vsfm]

COLMAP



COLMAP — COLMAP 3.9-dev documentation

https://colmap.github.io

COLMAP
3.9-dev

Search docs

- Installation
- Tutorial
- Database Format
- Camera Models
- Output Format
- Datasets
- Graphical User Interface
- Command-line Interface
- Frequently Asked Questions
- Changelog
- Contribution
- License
- Bibliography

/ COLMAP   View page source

COLMAP

Sparse model of central Rome using 21K photos produced by COLMAP's SfM pipeline.

Dense models of several landmarks produced by COLMAP's MVS pipeline.

About

COLMAP is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline with a graphical and command-line interface. It offers a wide range of features for reconstruction of ordered and unordered image collections. The software is licensed under the new BSD license. If you use this project for your research, please cite:

```
@inproceedings{schoenberger2016sfm,
    author={Sch\"{o}nberger, Johannes Lutz and Frahm, Jan-Michael},
```

Application: 3D from Internet Images

- Reconstruct 3D from unordered photo collections

[Building Rome in a Day, S. Agarwal et al 2009]