RANSAC: How many samples?

Let p_0 be the fraction of outliers (i.e., points on line)

Let n be the number of points needed to define hypothesis

$(n = 2$ for a line in the plane)

Suppose k samples are chosen

How many samples do we need to find a good solution?

13.7

RANSAC: How many samples?

Let p_0 be the fraction of outliers (i.e., points on line)

Let n be the numbe

($n = 2$ for a line

Suppose k samples

How many samples

13.7

$p(\text{inlier}) = 1 - p_0 = p_i$

$p(\text{correct sample}) = p_i^n$

$p(\text{no correct sample in } k \text{ trials}) = (1 - p_i^n)^k$

$(1 - p_i^n)^k < 0.01 = p_{fail}$

i.e. 99% chance of getting uncorrupted subset of points

$$k > \frac{\log 0.01}{\log (1 - p_i^n)}$$

e.g., $p_i = 0.5$, $n = 4$

50% inlier prob

homography

$$\underline{k} > \frac{-2}{\log 15/16} \simeq \underline{70}$$

RANSAC: How many samples?

Let p_0 be the fraction of outliers (i.e., points on line)

Let n be the numbe

($n = 2$ for a line

Suppose k samples

How many samples

13.7

$p(\text{inlier}) = 1 - p_0 = p_i$

Only approximately correct!

$p(\text{correct sample}) = p_i^n$

$p(\text{no correct sample in } k \text{ trials}) = (1 - p_i^n)^k$

$(1 - p_i^n)^k < 0.01 = p_{fail}$ i.e. 99% chance of getting uncorrupted subset of points

$$k > \frac{\log 0.01}{\log(1 - p_i^n)}$$

e.g., $p_i = 0.5$, $n = 4$

50% inlier prob, homography

$$\underline{k >} \frac{-2}{\log {}^{15}/_{16}} \simeq \underline{70}$$

RANSAC: How many samples? (Exact)

The original RANSAC paper [19] suggested to use

$$P_a = p^k \tag{3}$$

for the all-inlier probability P, where p is the inlier ratio and k the number of sampled measurements. This has since become the standard approach for computing the required number of iterations in RANSAC.

However, this only provides the approximate probability P_a, as drawing an inlier measurement for our sample changes the inlier ratio when sampling another measurement in the same iteration. Using uniform random sampling, the exact probability P_e can be computed as the ratio between the number of all-inlier samples and the number of possible samples, *i.e.*

$$P_e = \frac{\binom{pn}{k}}{\binom{n}{k}} \tag{4}$$

or equivalently formulated as

$$P_e = \prod_{i=0}^{k-1} \frac{pn - i}{n - i} \quad \text{if } pn \geq k \text{ and } 0 \text{ otherwise,} \tag{5}$$

RANSAC: How many samples? (Exact)

The original RANSAC paper [19] suggested to use

$$P_a = p^k \tag{3}$$

for the all-inlier probability P, where p is the inlier ratio and k the number of sampled measurements. This has since become the standard approach for computing the required number of iterations in RANSAC.

However, this only provides the approximate probability P_a, as drawing an inlier measurement for our sample changes the inlier ratio when sampling another measurement in the same iteration. Using uniform random sampling, the exact probability P_e can be computed as the ratio between the number of all-inlier samples and the number of possible samples, *i.e.*

$$P_e = \frac{\binom{pn}{k}}{\binom{n}{k}} \tag{4}$$

or equivalently formulated as

$$P_e = \prod_{i=0}^{k-1} \frac{pn - i}{n - i} \quad \text{if } pn \geq k \text{ and 0 otherwise,} \tag{5}$$

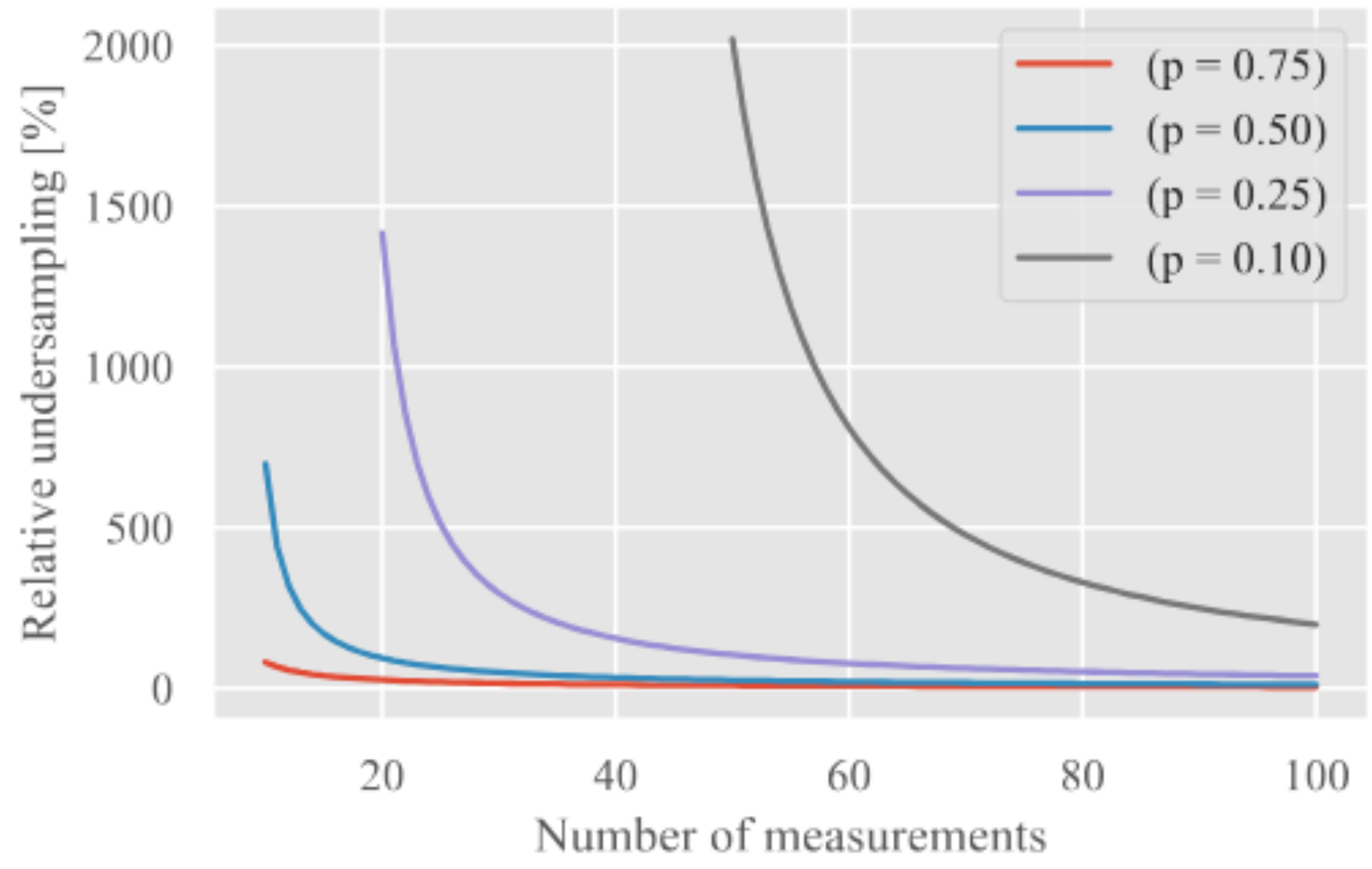


Figure 4. **Amount of undersampling.** The graph shows the relative difference of required iterations for the exact versus the approximate stopping criterion as $N_{e/a} = \frac{N_e - N_a}{N_a}$ to reach a target success probability $s = 0.99$ with a sample size $k = 5$. The approximation leads to severe undersampling in low inlier scenarios.

RANSAC: How many samples? (Exact)

The original RANSAC paper [19] suggested to use

$$P_a = p^k \tag{3}$$

for the all-inlier probability P, where p is the inlier ratio and k the number of sampled measurements. This has since become the standard approach for computing the required number of iterations in RANSAC.

However, this only provides the approximate probability P_a, as drawing an inlier measurement for our sample changes the inlier ratio when sampling another measurement in the same iteration. Using uniform random sampling, the exact probability P_e can be computed as the ratio between the number of all-inlier samples and the number of possible samples, *i.e.*

$$P_e = \frac{\binom{pn}{k}}{\binom{n}{k}} \tag{4}$$

or equivalently formulated as

$$P_e = \prod_{i=0}^{k-1} \frac{pn - i}{n - i} \quad \text{if } pn \geq k \text{ and } 0 \text{ otherwise,} \tag{5}$$

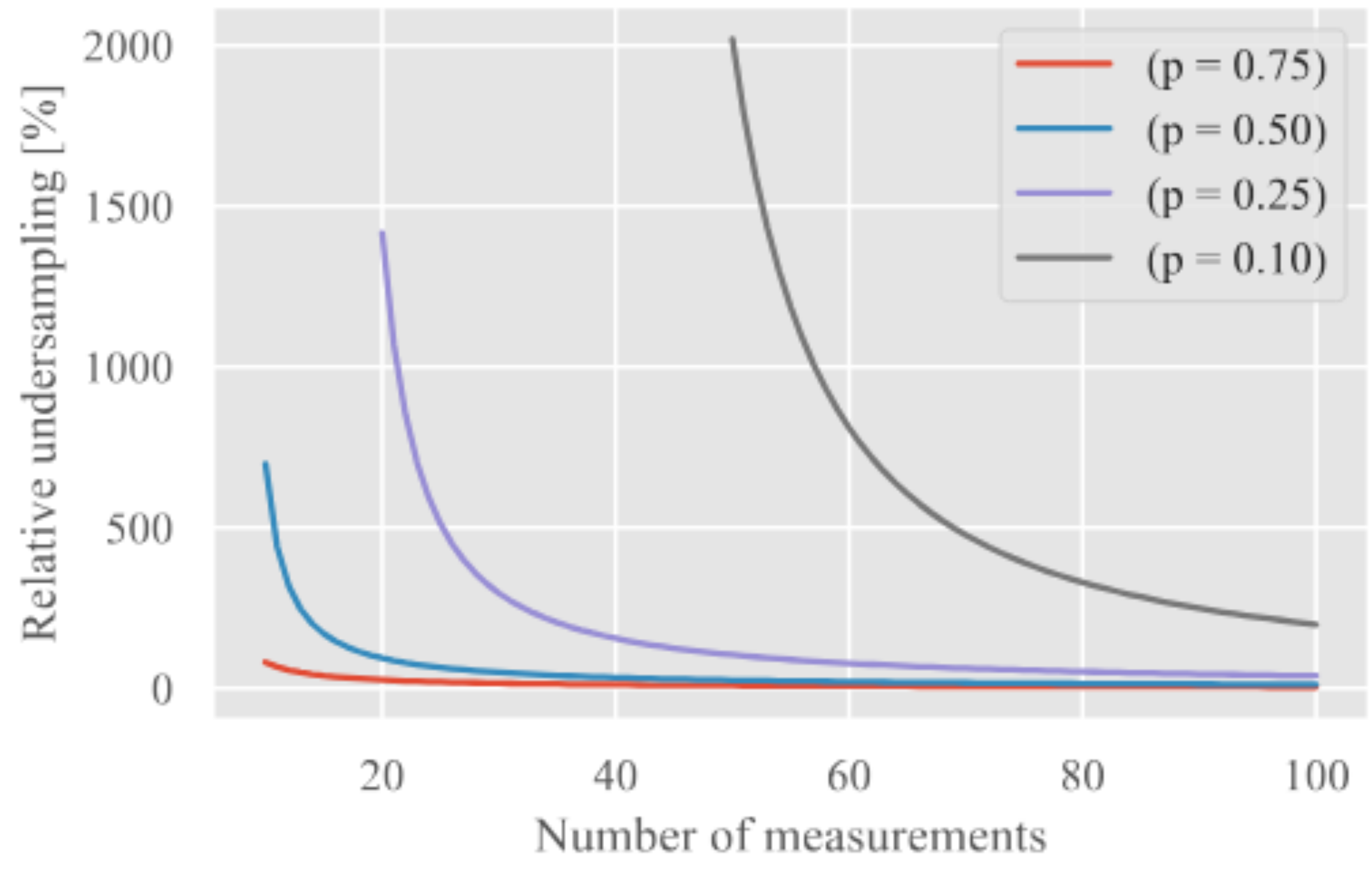


Figure 4. **Amount of undersampling.** The graph shows the relative difference of required iterations for the exact versus the approximate stopping criterion as $N_{e/a} = \frac{N_e - N_a}{N_a}$ to reach a target success probability $s = 0.99$ with a sample size $k = 5$. The approximation leads to severe undersampling in low inlier scenarios.

How bad practically?

RANSAC: How many samples? (Exact)

Parameters		AUC@5			AUC@10			AUC@20			Average Runtime	
n	s	Approx.	Exact	Δ	Approx.	Exact	Δ	Approx.	Exact	Δ	Approx.	Exact
Homography												
20	0.95	2.99	3.31	+10.70%	5.87	6.48	+10.39%	9.94	10.91	+9.76%	0.2ms	0.7ms
	0.99	3.25	3.51	+8.00%	6.23	6.74	+8.19%	10.53	11.27	+7.03%	0.3ms	0.7ms
100	0.95	10.03	10.03	+0.00%	17.32	17.33	+0.06%	25.19	25.20	+0.04%	1.0ms	1.0ms
	0.99	10.10	10.13	+0.30%	17.51	17.53	+0.11%	25.39	25.39	+0.00%	1.0ms	1.0ms
200	0.95	11.69	11.70	+0.09%	19.79	19.80	+0.05%	28.18	28.18	+0.00%	1.2ms	1.2ms
	0.99	11.72	11.72	+0.00%	19.82	19.82	+0.00%	28.14	28.14	+0.00%	1.2ms	1.2ms
1000	0.95	58.49	58.49	+0.00%	76.44	76.44	+0.00%	88.21	88.21	+0.00%	3.9ms	4.0ms
	0.99	58.78	58.78	+0.00%	76.55	76.55	+0.00%	88.27	88.27	+0.00%	4.3ms	4.3ms
Essential matrix												
20	0.95	25.27	25.87	+2.37%	38.63	39.42	+2.05%	52.15	53.02	+1.67%	0.4ms	0.8ms
	0.99	26.08	26.63	+2.11%	39.66	40.36	+1.77%	53.26	54.01	+1.41%	0.6ms	1.4ms
100	0.95	45.78	45.93	+0.33%	60.81	61.00	+0.31%	73.11	73.29	+0.25%	2.0ms	2.1ms
	0.99	46.77	46.89	+0.26%	61.86	62.01	+0.24%	74.07	74.21	+0.19%	2.1ms	2.2ms
200	0.95	49.89	49.95	+0.12%	64.39	64.47	+0.12%	76.00	76.06	+0.08%	2.2ms	2.2ms
	0.99	50.73	50.76	+0.06%	65.38	65.41	+0.05%	76.91	76.94	+0.04%	2.3ms	2.3ms
1000	0.95	37.88	37.87	+-0.03%	51.83	51.83	+0.00%	63.18	63.18	+0.00%	1.8ms	1.8ms
	0.99	38.19	38.19	+0.00%	52.48	52.48	+0.00%	63.97	63.96	+-0.02%	2.0ms	2.0ms
Fundamental matrix												
20	0.95	8.59	9.02	+5.01%	15.98	16.67	+4.32%	26.50	27.44	+3.55%	0.2ms	0.5ms
	0.99	8.98	9.36	+4.23%	16.60	17.22	+3.73%	27.35	28.17	+3.00%	0.3ms	0.7ms
100	0.95	24.62	24.79	+0.69%	37.32	37.56	+0.64%	50.97	51.24	+0.53%	1.4ms	1.4ms
	0.99	25.38	25.54	+0.63%	38.35	38.57	+0.57%	52.11	52.36	+0.48%	1.5ms	1.5ms
200	0.95	30.09	30.16	+0.23%	43.54	43.63	+0.21%	57.08	57.17	+0.16%	1.9ms	1.9ms
	0.99	30.86	30.93	+0.23%	44.54	44.62	+0.18%	58.22	58.31	+0.15%	3.1ms	2.0ms
1000	0.95	26.37	26.37	+0.00%	40.42	40.42	+0.00%	54.85	54.85	+0.00%	2.2ms	2.2ms
	0.99	26.57	26.58	+0.04%	40.44	40.46	+0.05%	54.37	54.38	+0.02%	2.3ms	2.3ms

Table 4. **Relative camera pose estimation.** Two-view homography ($k = 4$), essential matrix ($k = 5$), fundamental matrix estimation ($k = 7$) results with varying number of measurements n and target success probability s reported as end-to-end AUC (higher is better) with relative improvements Δ and runtime metrics.

Recap

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

15.2

let $e = 0$ for a single pixel

$$I_t(x) - I_{t+\Delta t}(x) = \frac{\partial I_1}{\partial x}^T \Delta x$$

$\div \Delta t$

$$-\frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}^T \frac{\Delta x}{\Delta t}$$

velocity

temporal difference

$$I_t + \frac{\partial I}{\partial x}^T \frac{\partial x}{\partial t} = 0$$

$$I_t + \begin{pmatrix} I_x \\ I_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix} = 0$$

$$\underline{I_t + I_x v_x + I_y v_y = 0}$$

optical flow constraint eqn (single patch)

Optical Flow in **1D**

Consider a 1D function moving at velocity v

15.3

<u>15.3</u>

$$I_t + \begin{pmatrix} I_x \\ I_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix} = 0 \qquad I_t + \nabla I \cdot \underline{v} = 0$$

Optical flow in 1D

$$I(x, t+\Delta t) = I(x, t) - \Delta x \frac{\partial I}{\partial x}$$

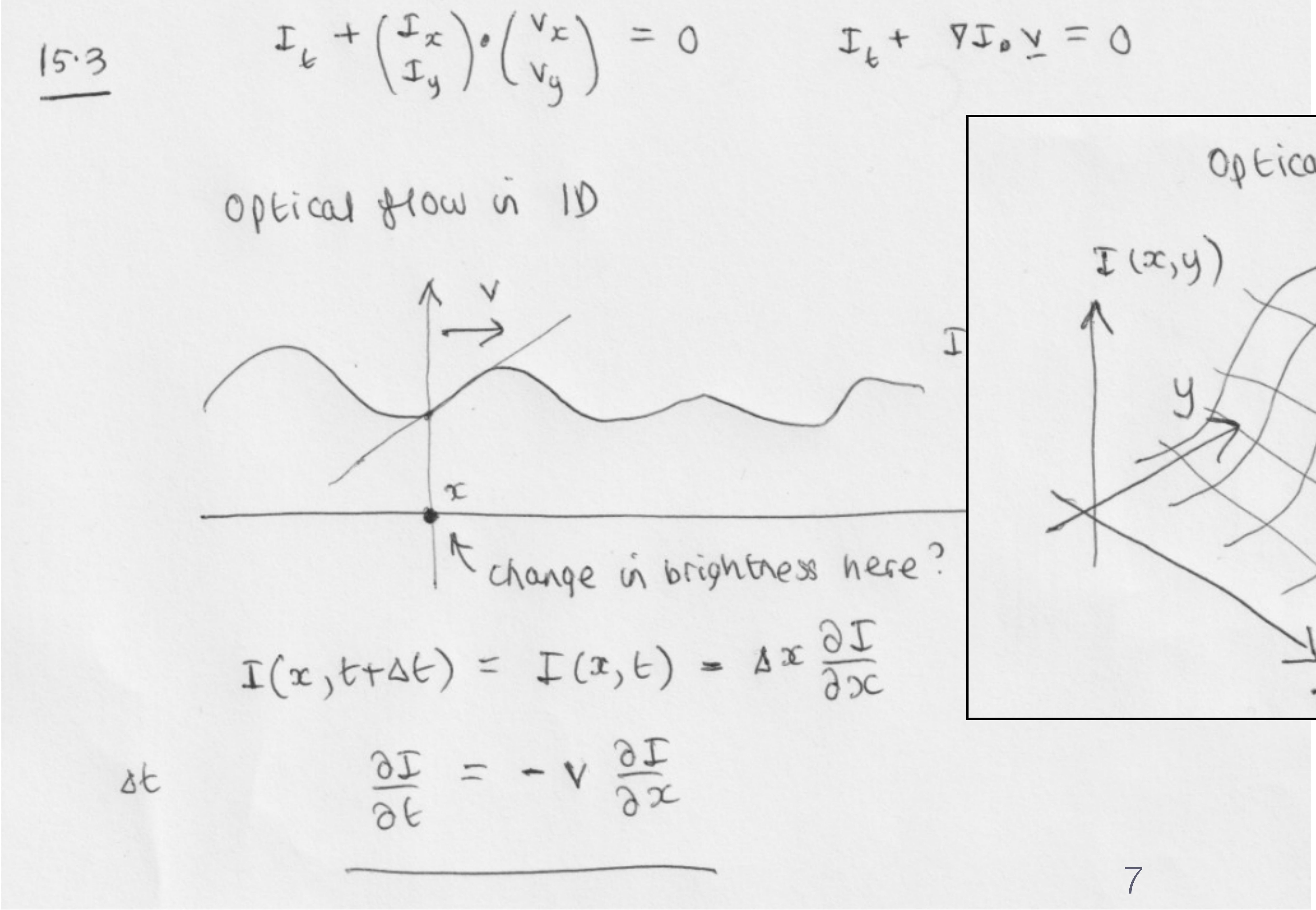
Δt

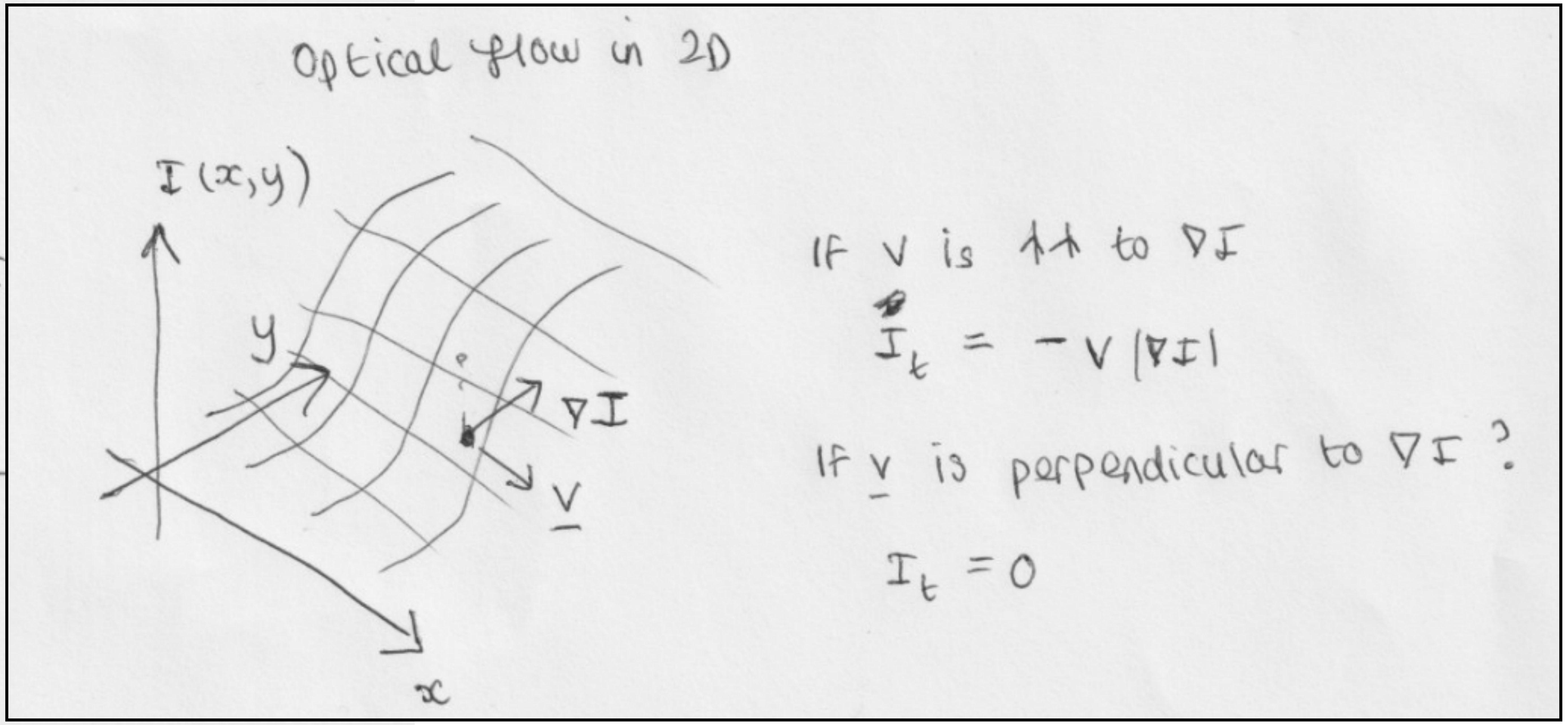
$$\frac{\partial I}{\partial t} = -v \frac{\partial I}{\partial x}$$

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

Assumption: Locally **constant** motion

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

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

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

Horn-Schunck Optical Flow

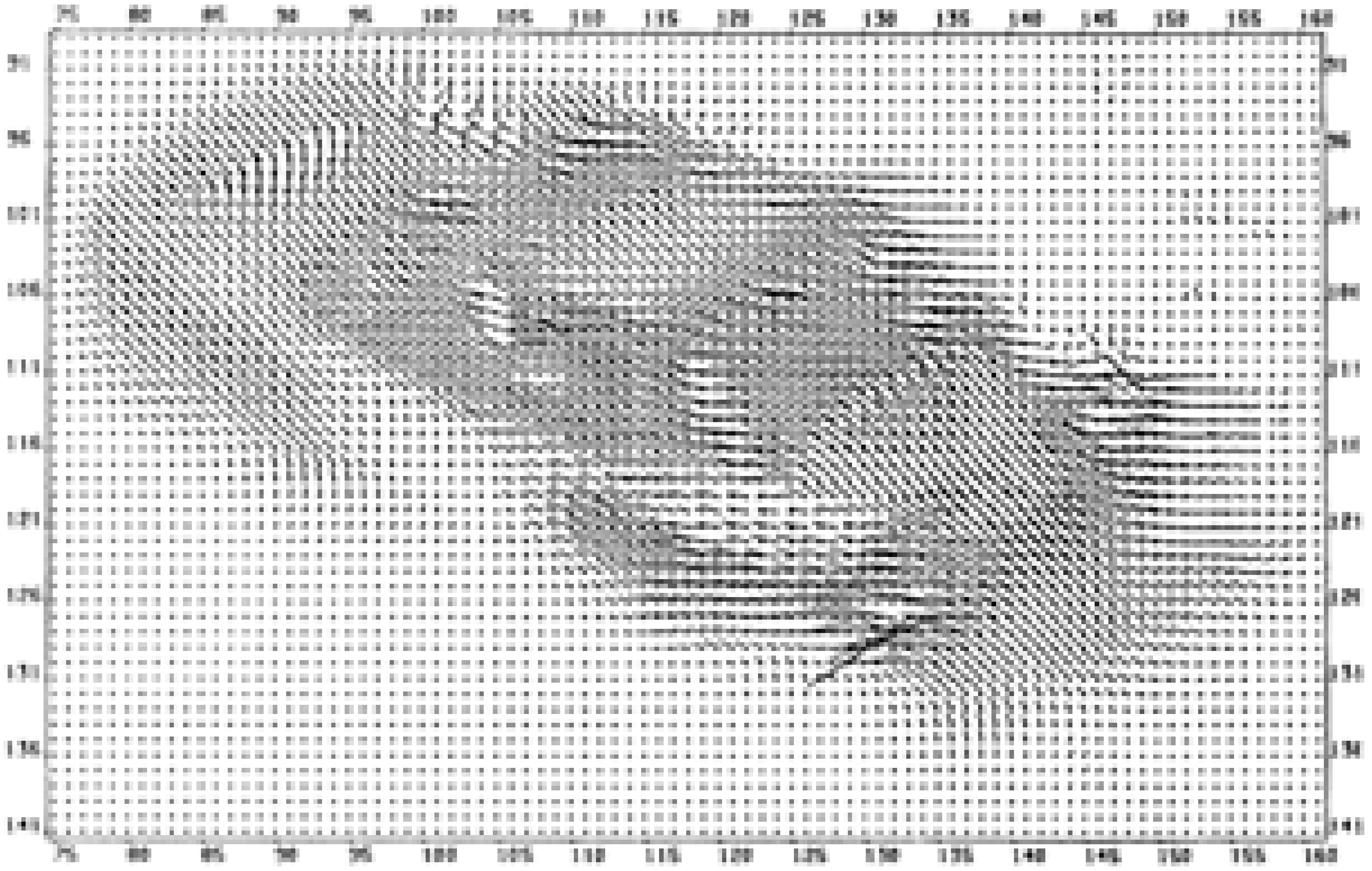
Assumption: Locally **smooth** motion

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.

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

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]

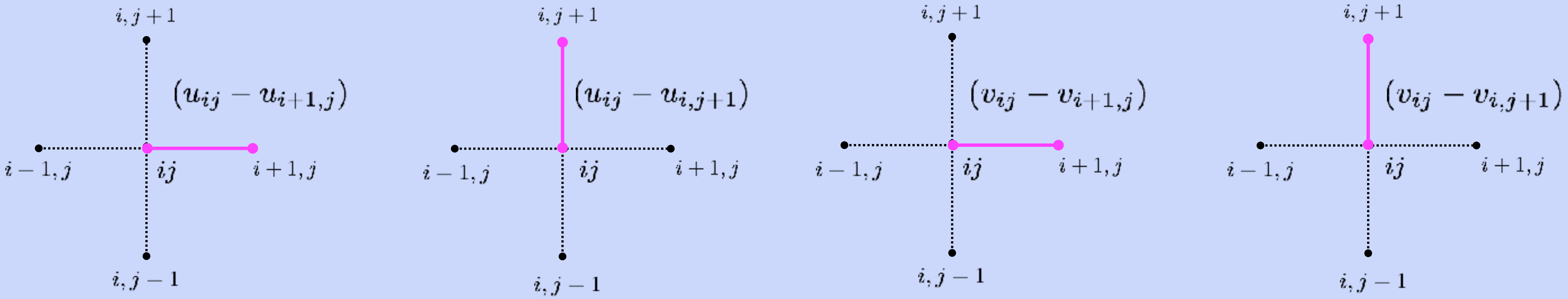
Horn-Schunck Optical Flow

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)

Optical Flow **Summary**

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?

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

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**

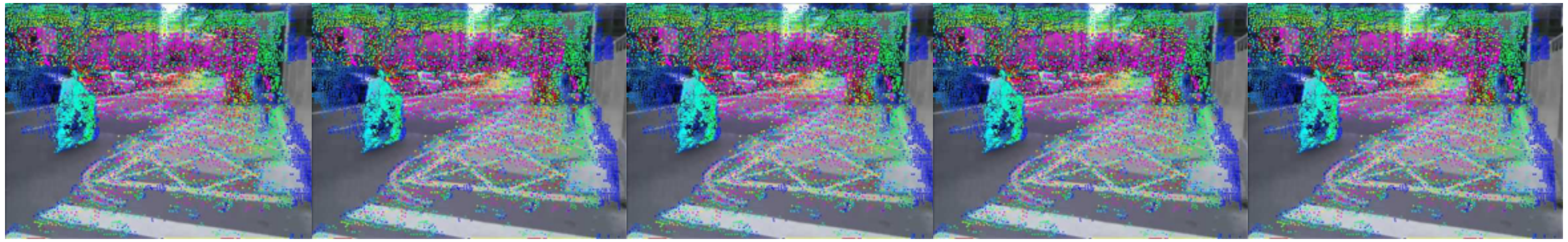
$$I_x u + I_y v + I_t = 0$$

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

THE UNIVERSITY OF BRITISH COLUMBIA

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**

Learning **Goals**

Putting it all together

2-view **Rigid** Matching

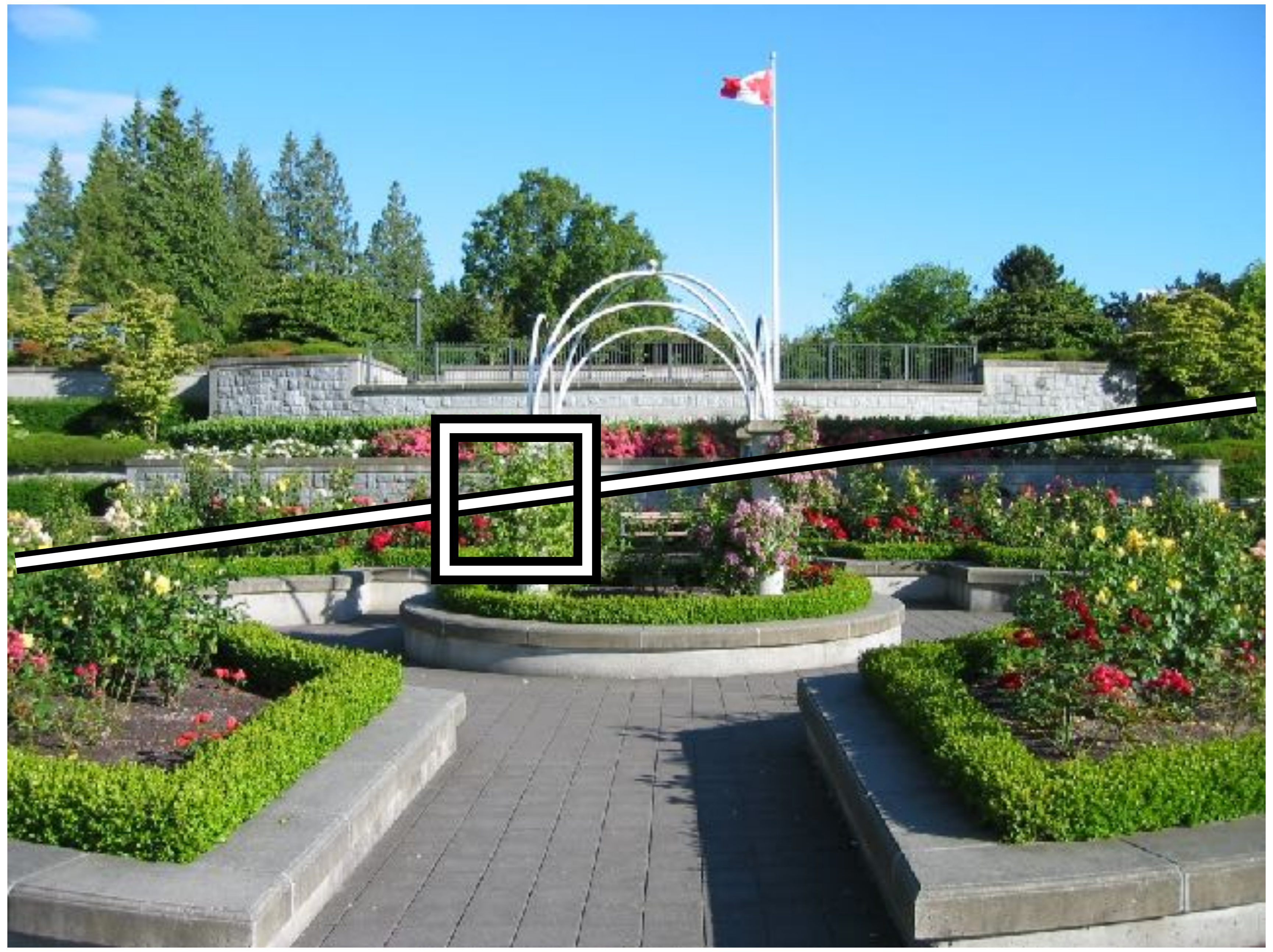
1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines



2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

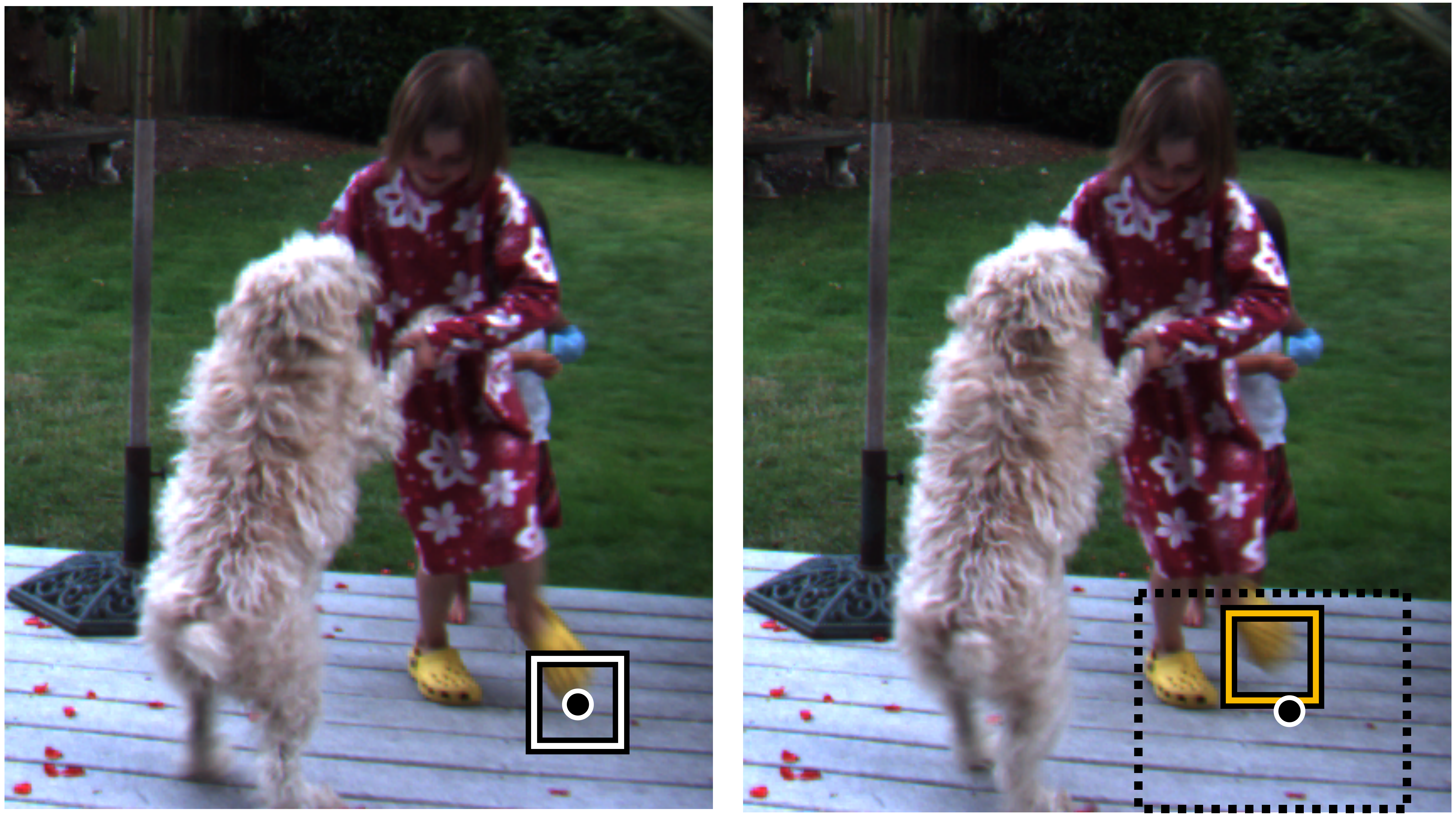
1D search, points constrained to lie along epipolar lines

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Non-Rigid** Matching

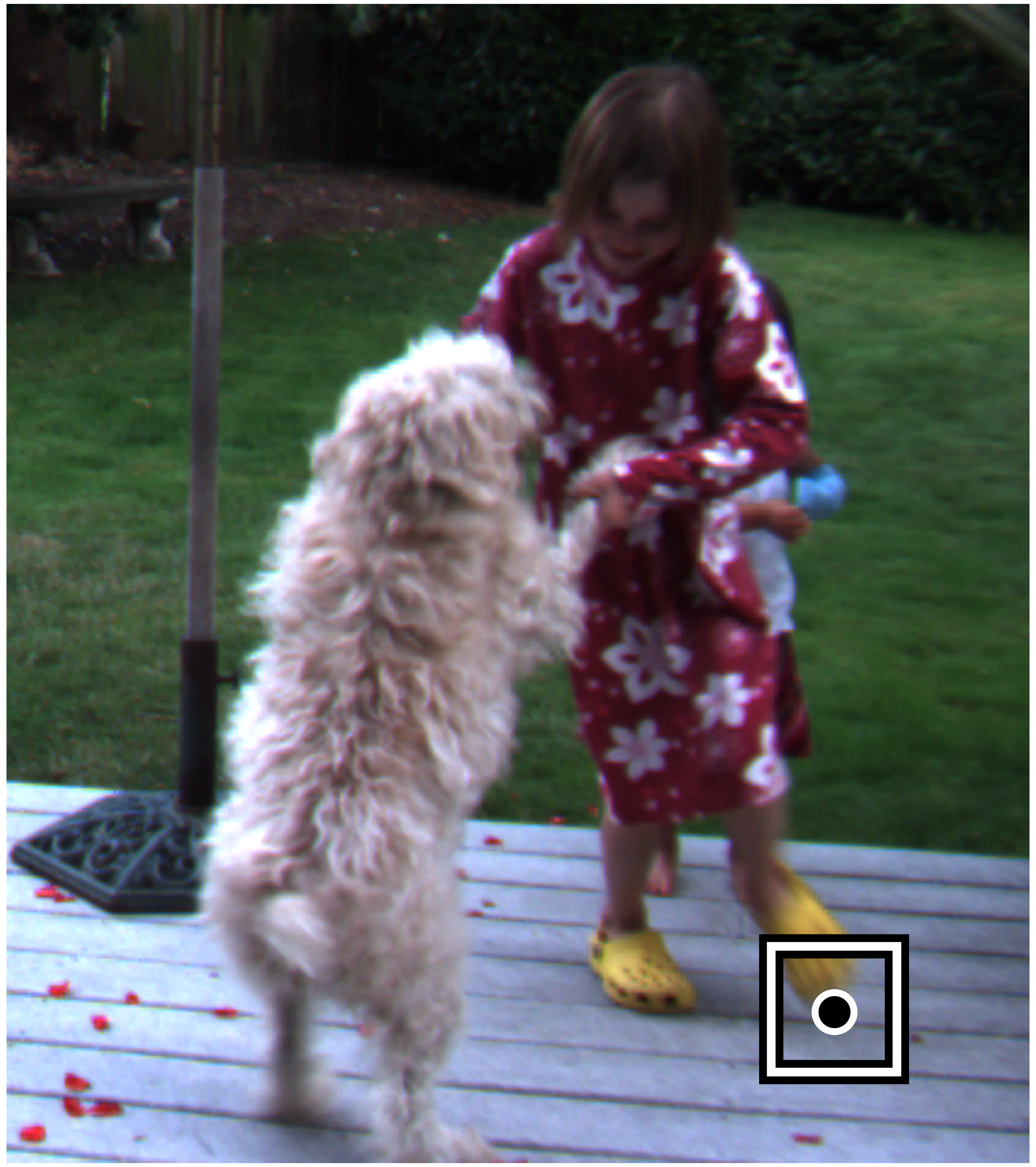
2D search, points can move anywhere in the image



[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

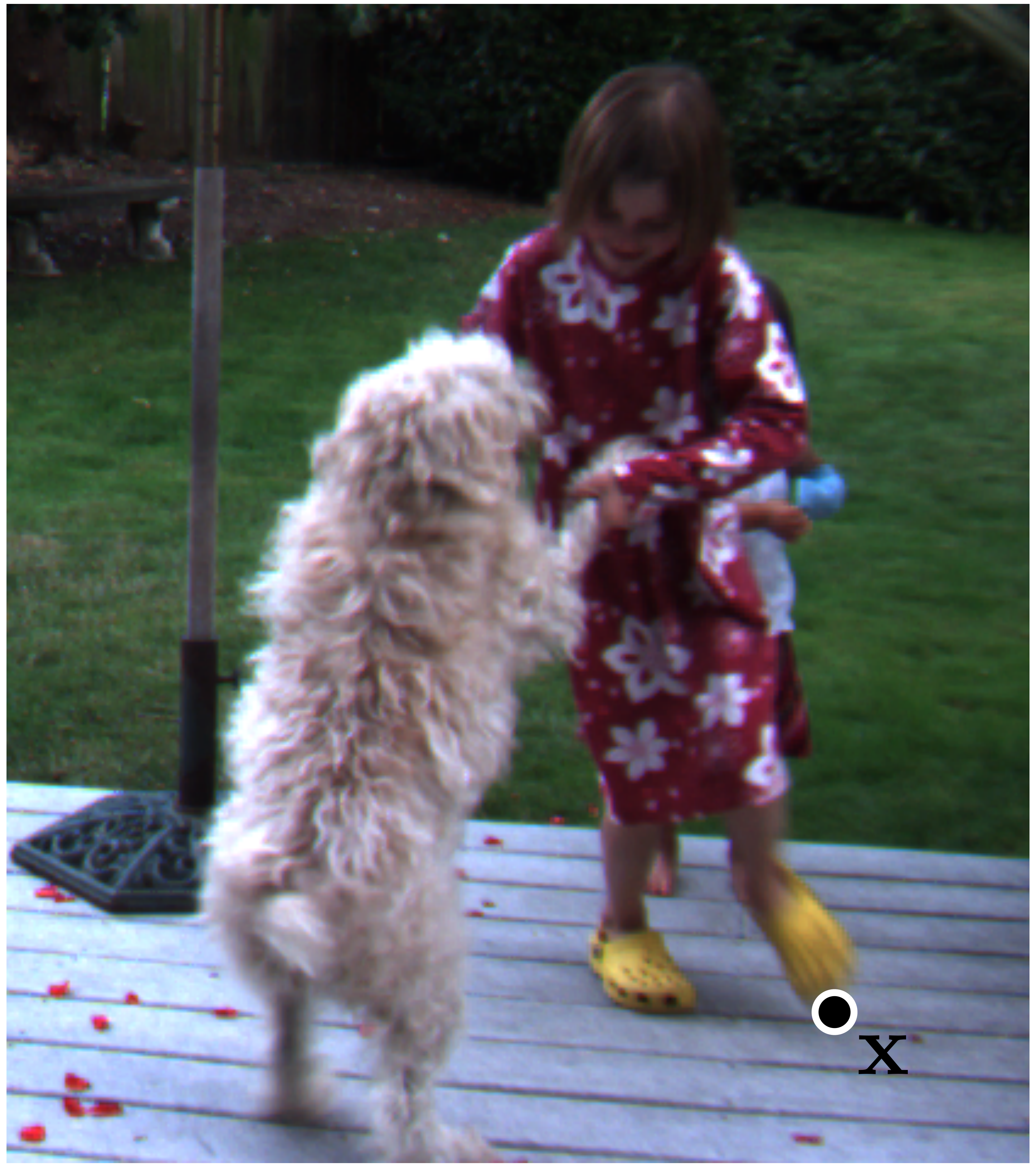
2D search, points can move anywhere in the image

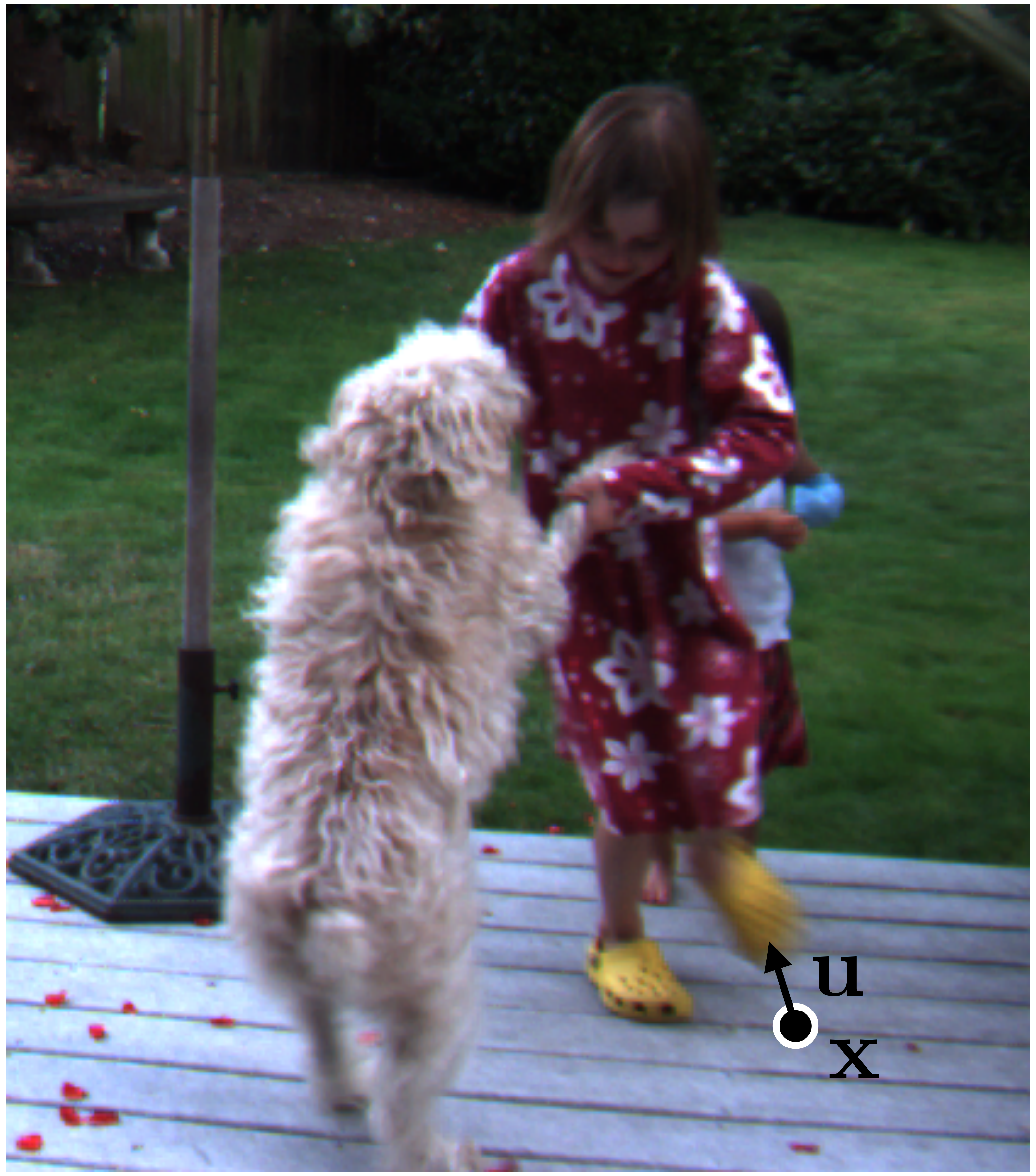


[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

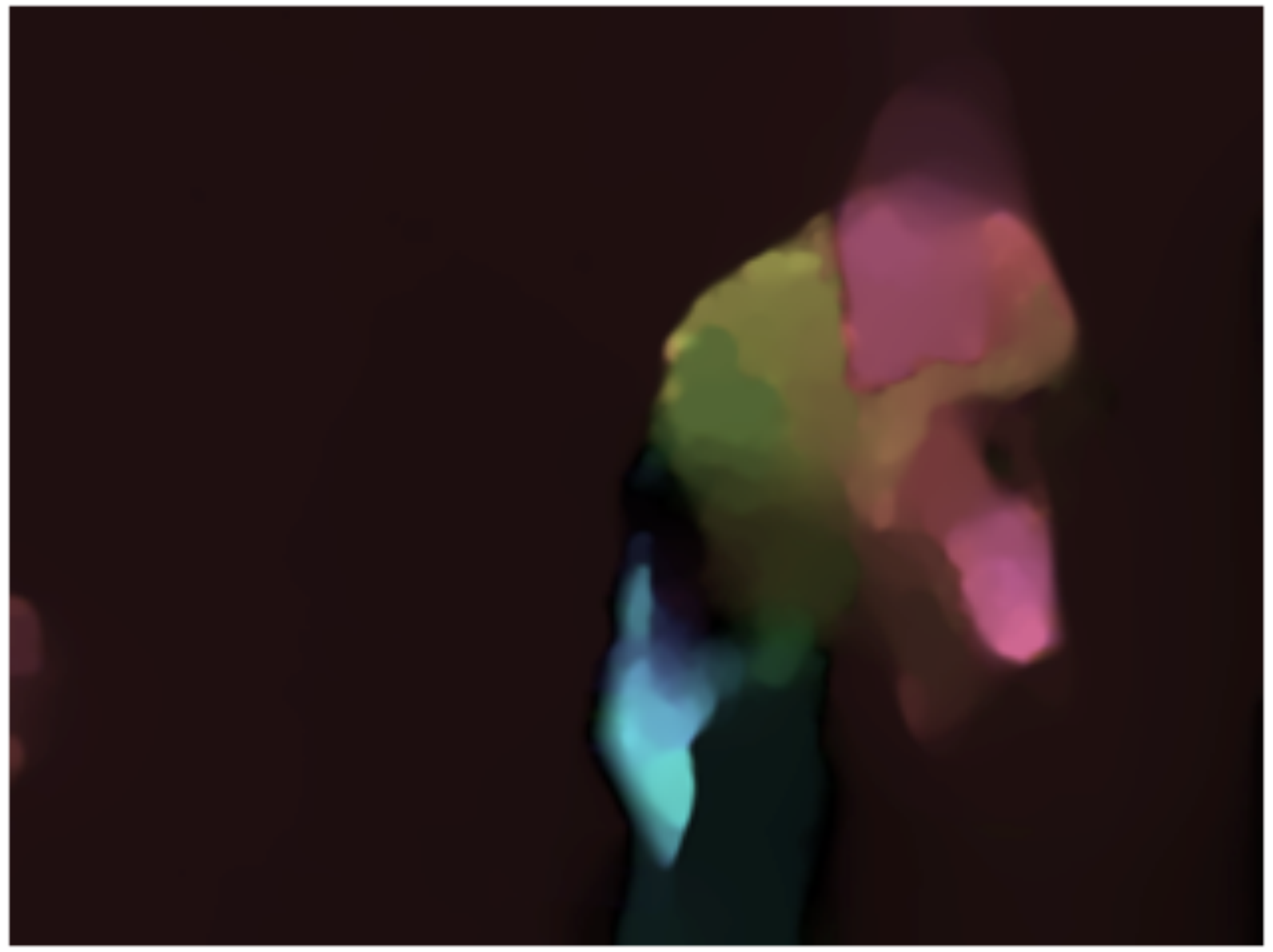
2D search, points can move anywhere in the image

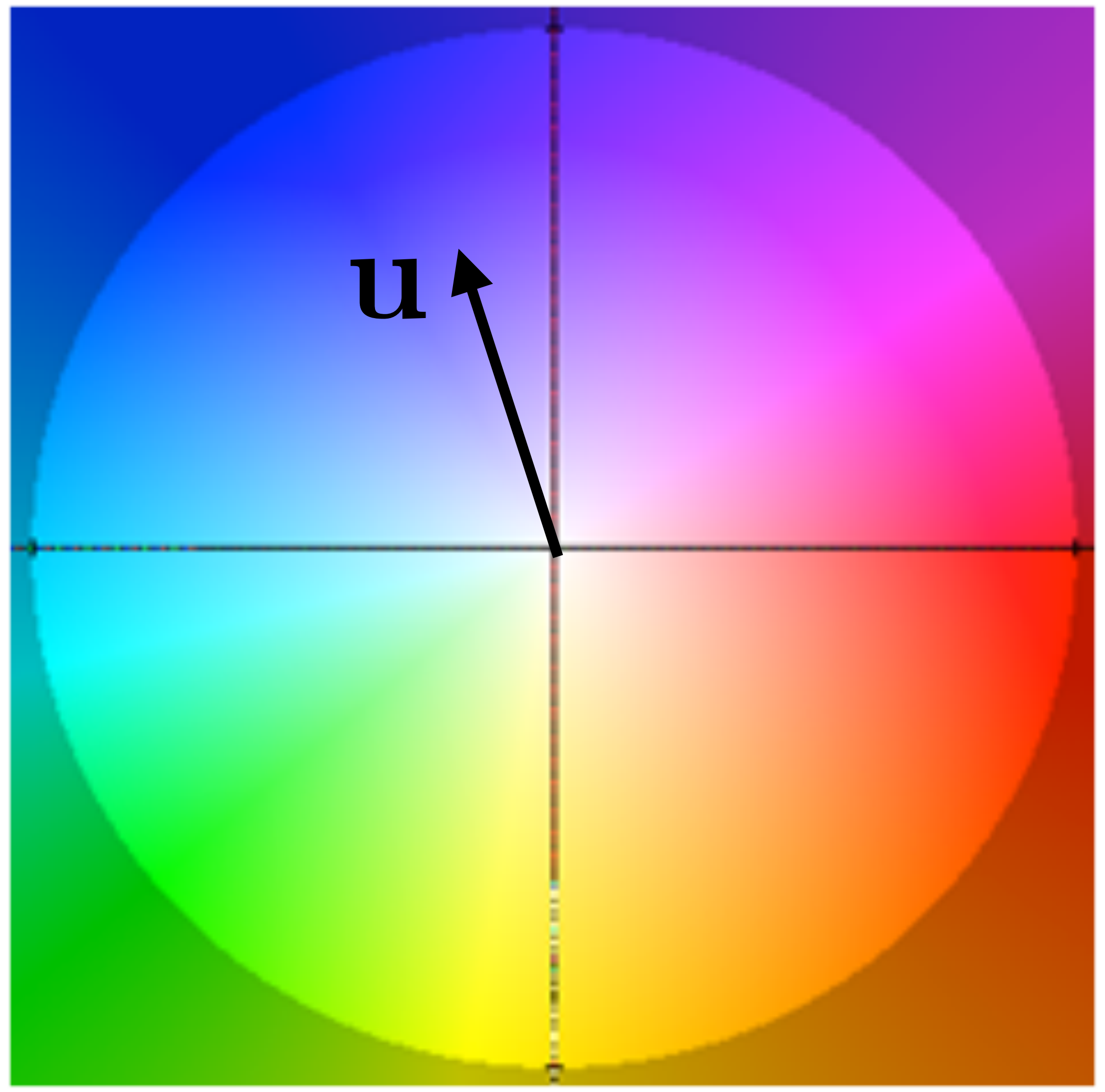




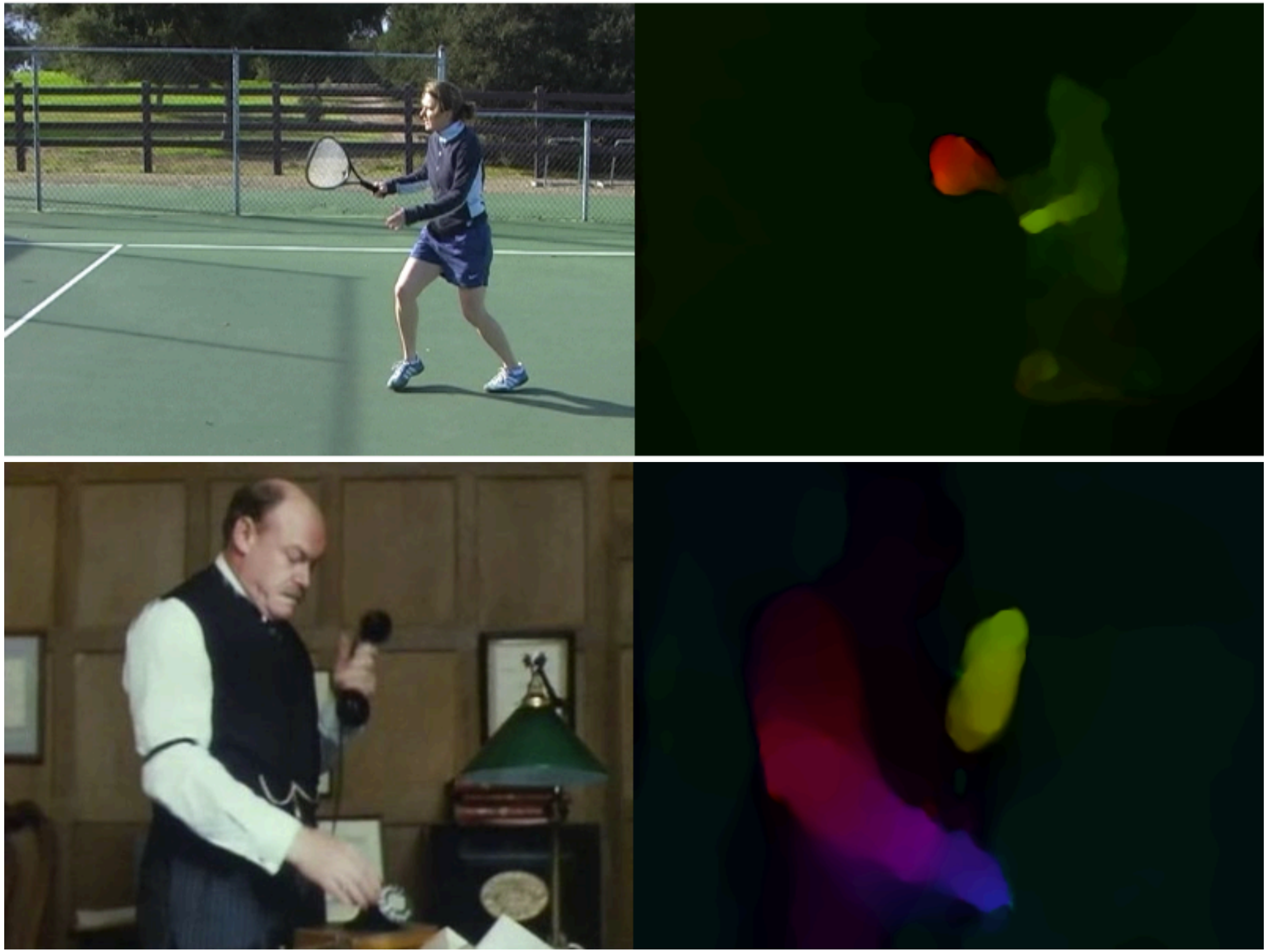
[vision.middlebury.edu/flow]

Optical Flow: Example 1





Optical Flow: Example 2



[Brox Malik 2011]

Multiview + Sparse SFM

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

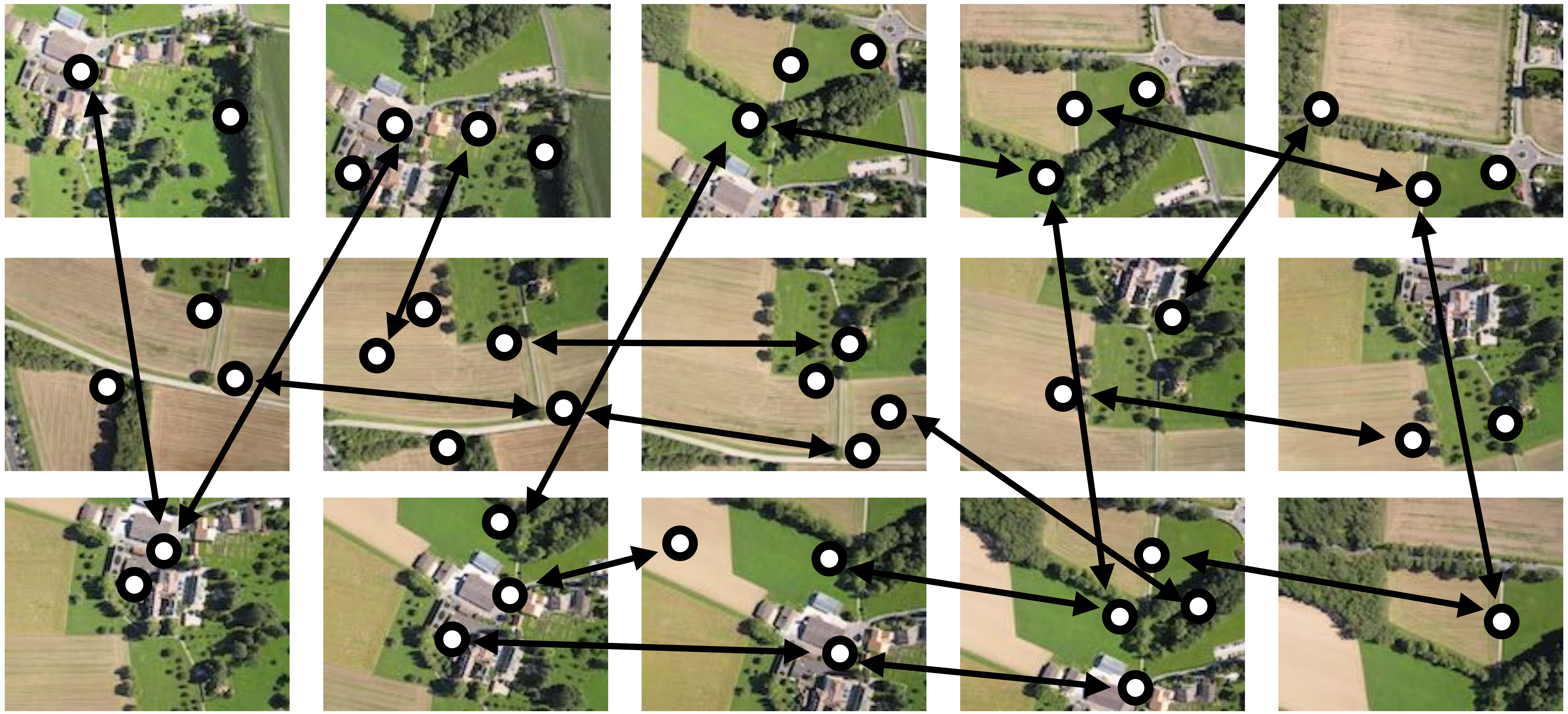
[Szeliski 11.4]

Multiview Image Alignment

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

Multiview Image Alignment

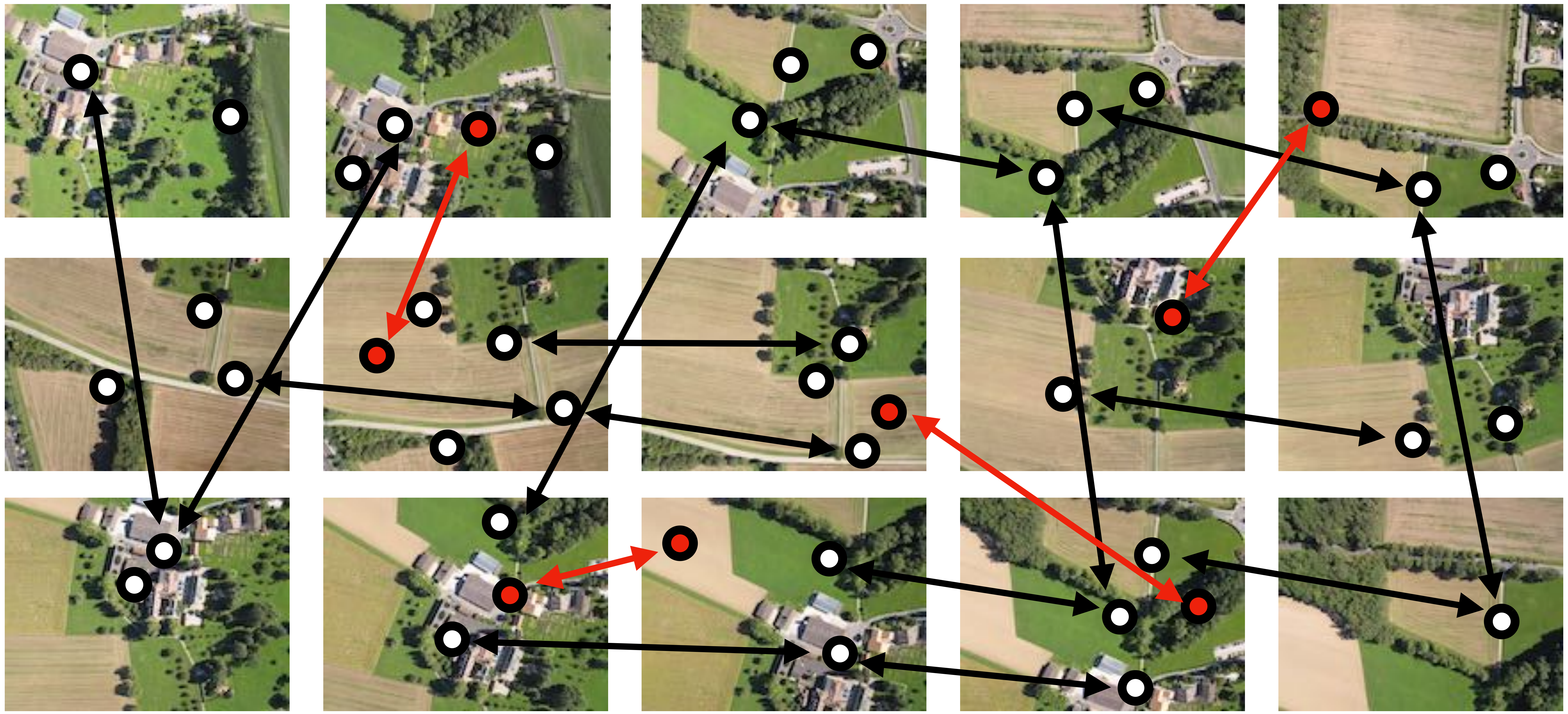
Align a set of images given a motion model (e.g., planar affine)



Step 1: Find all matches between images using SIFT

Multiview Image Alignment

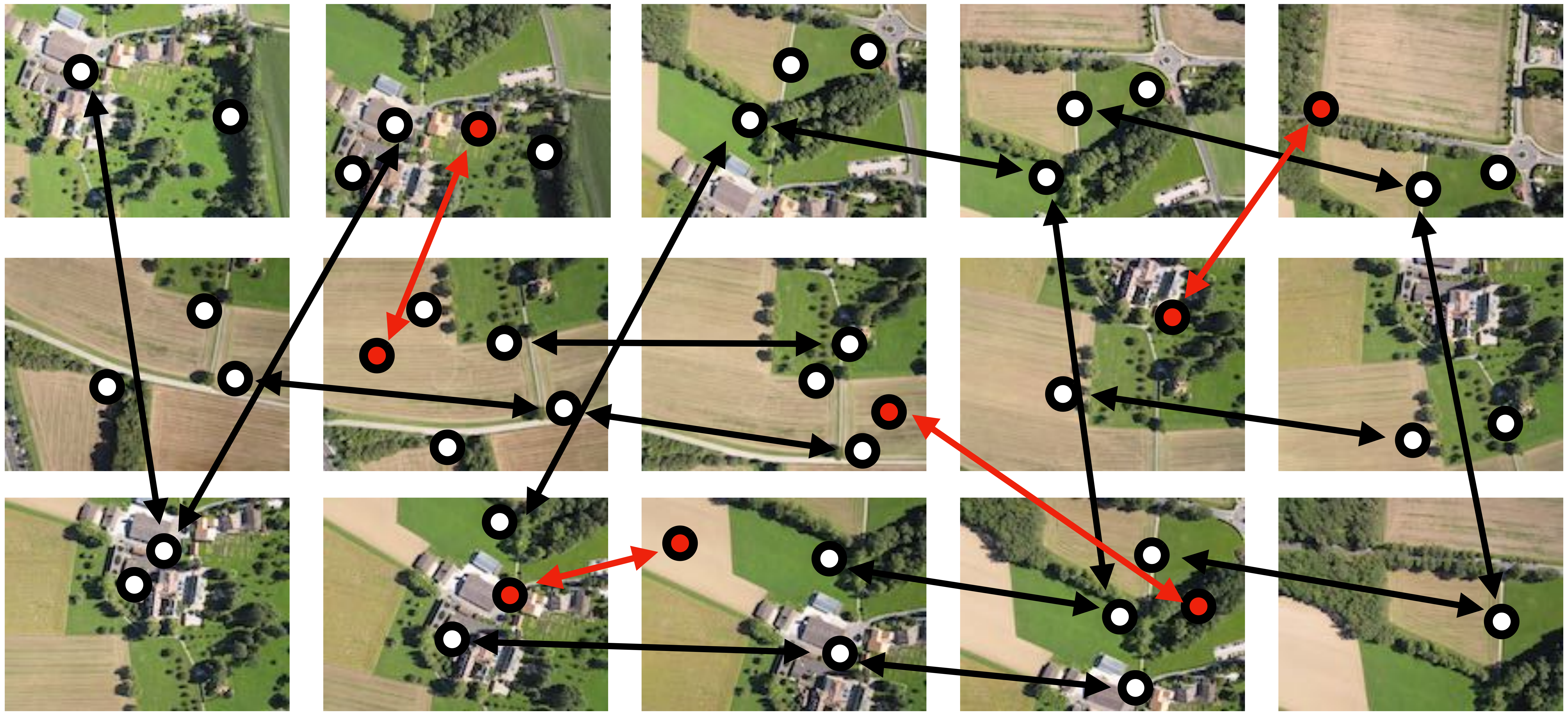
Align a set of images given a motion model (e.g., planar affine)



Step 1: Find all matches between images using SIFT

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

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)

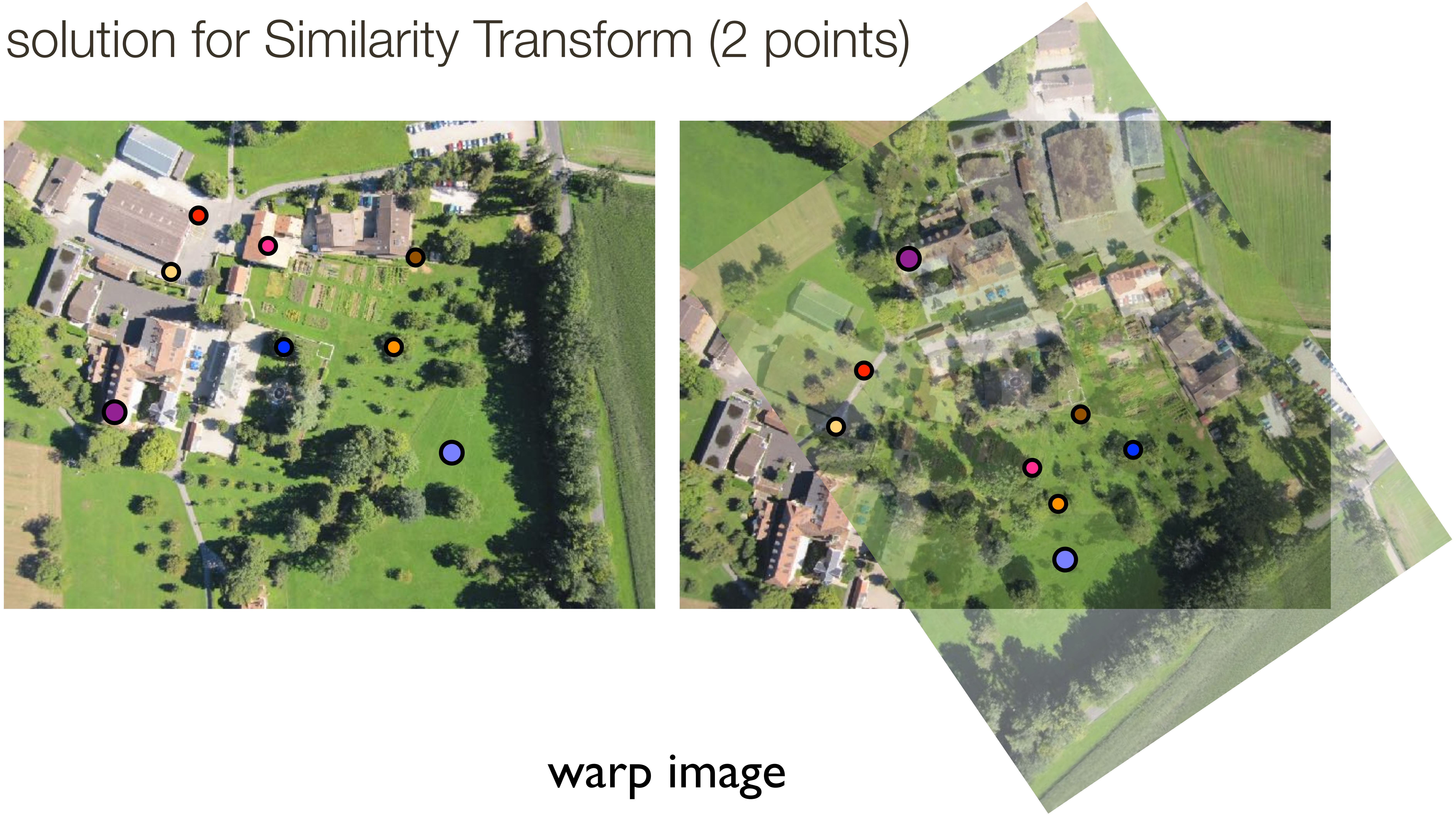
Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

choose light blue, purple

Recap: Image **Alignment + RANSAC**

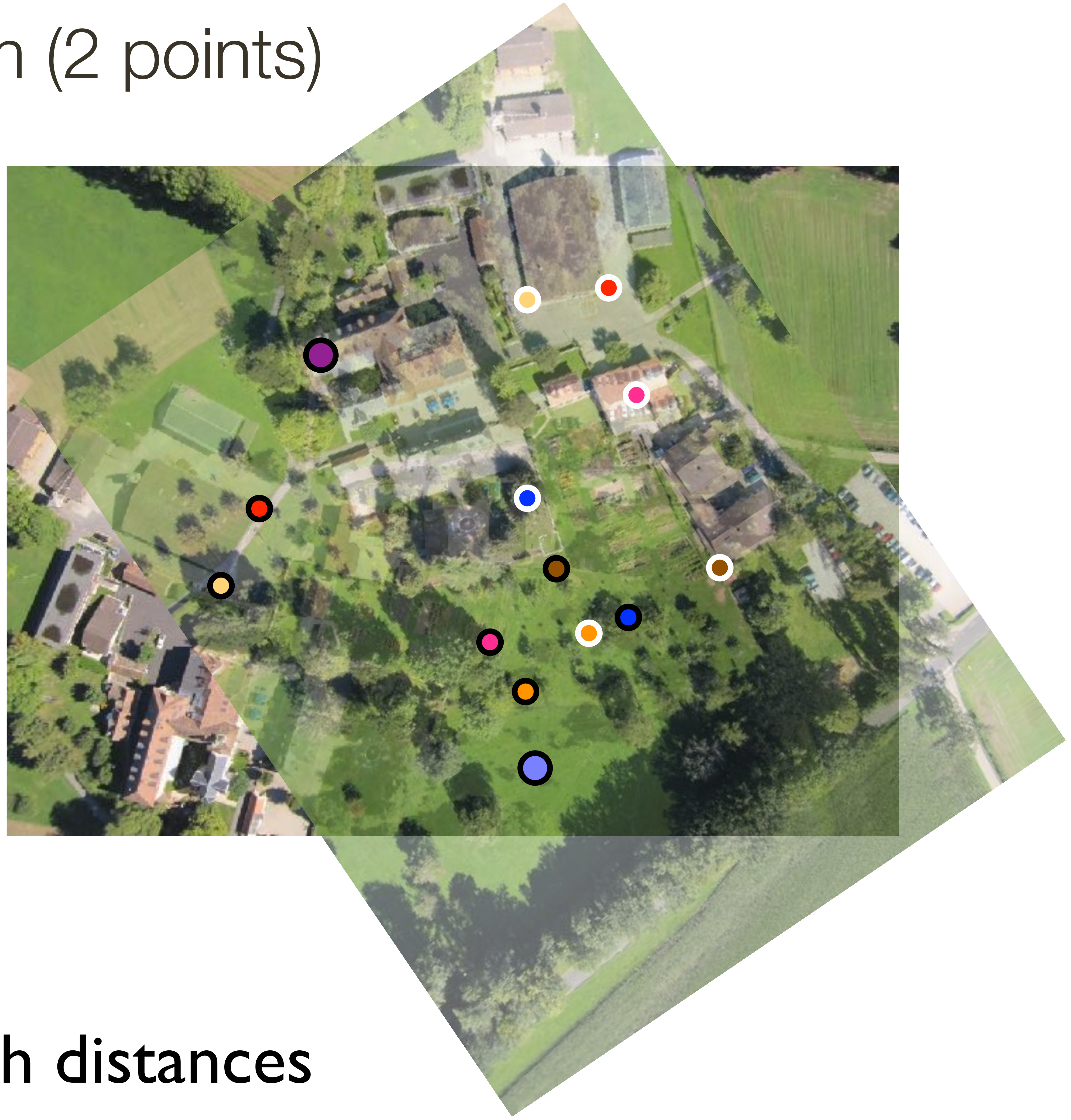
RANSAC solution for Similarity Transform (2 points)



warp image

Recap: Image **Alignment + RANSAC**

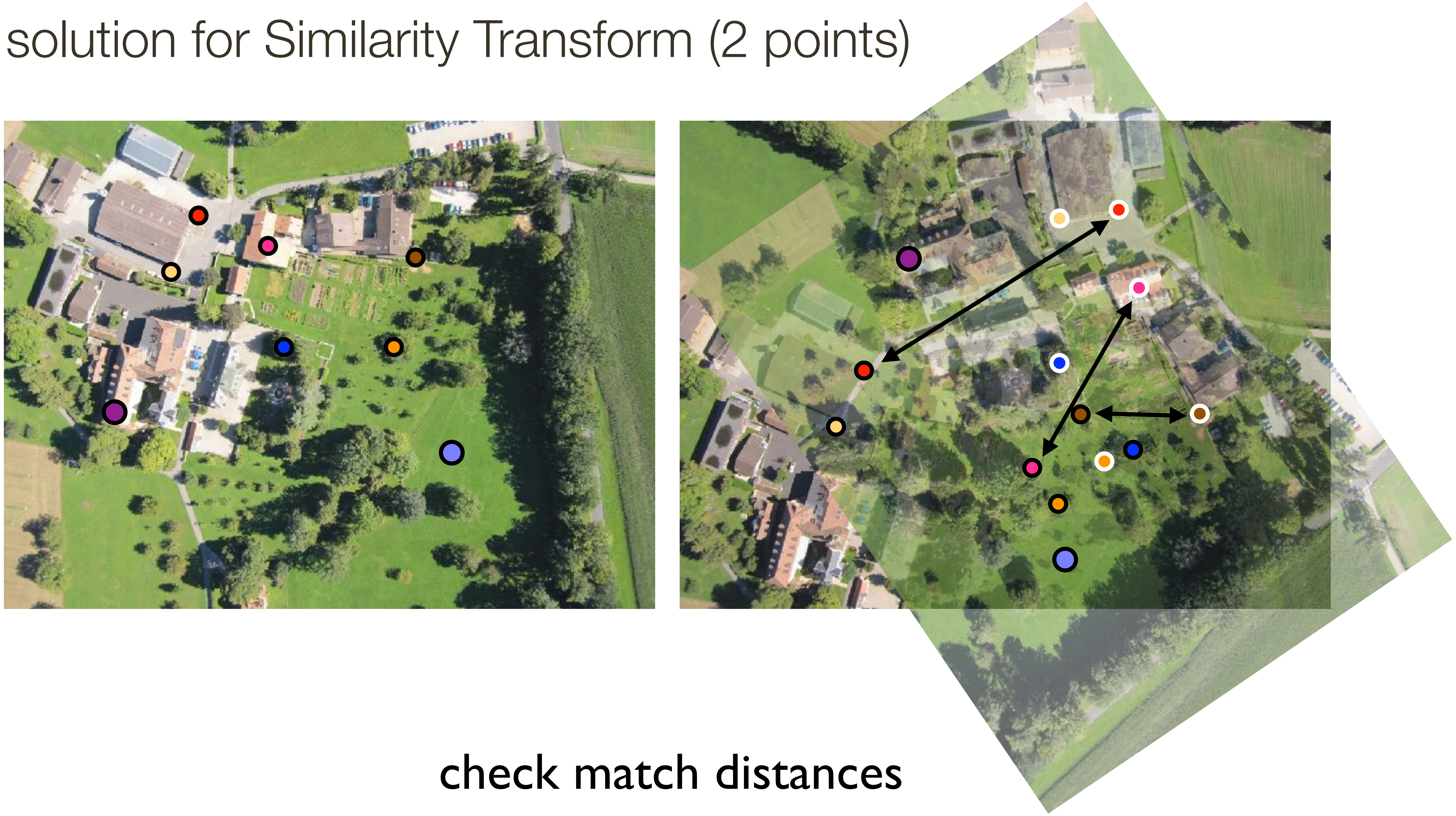
RANSAC solution for Similarity Transform (2 points)



check match distances

Recap: Image **Alignment + RANSAC**

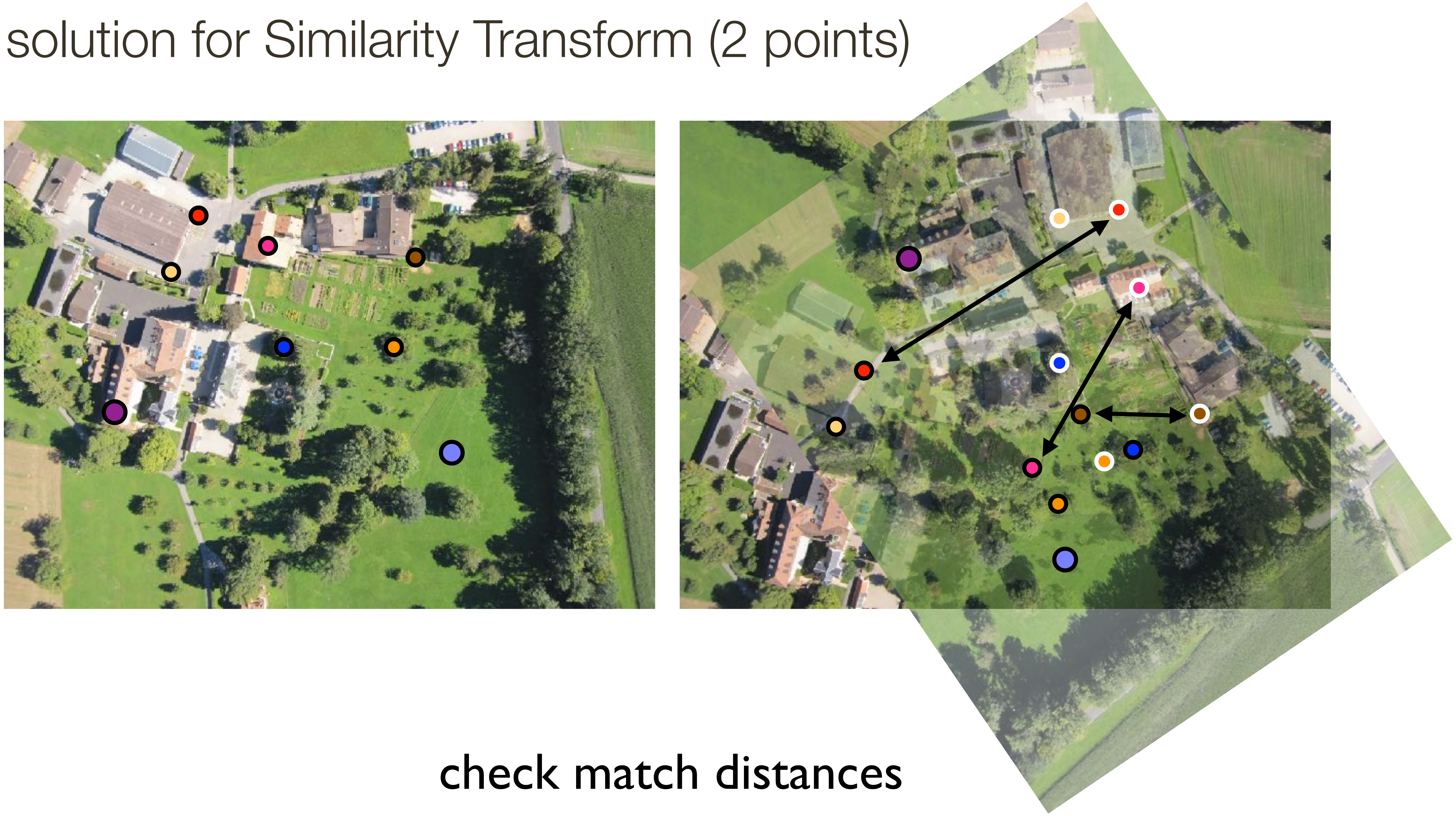
RANSAC solution for Similarity Transform (2 points)



check match distances

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)



check match 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 pink, blue

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

warp image

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match 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

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

warp image

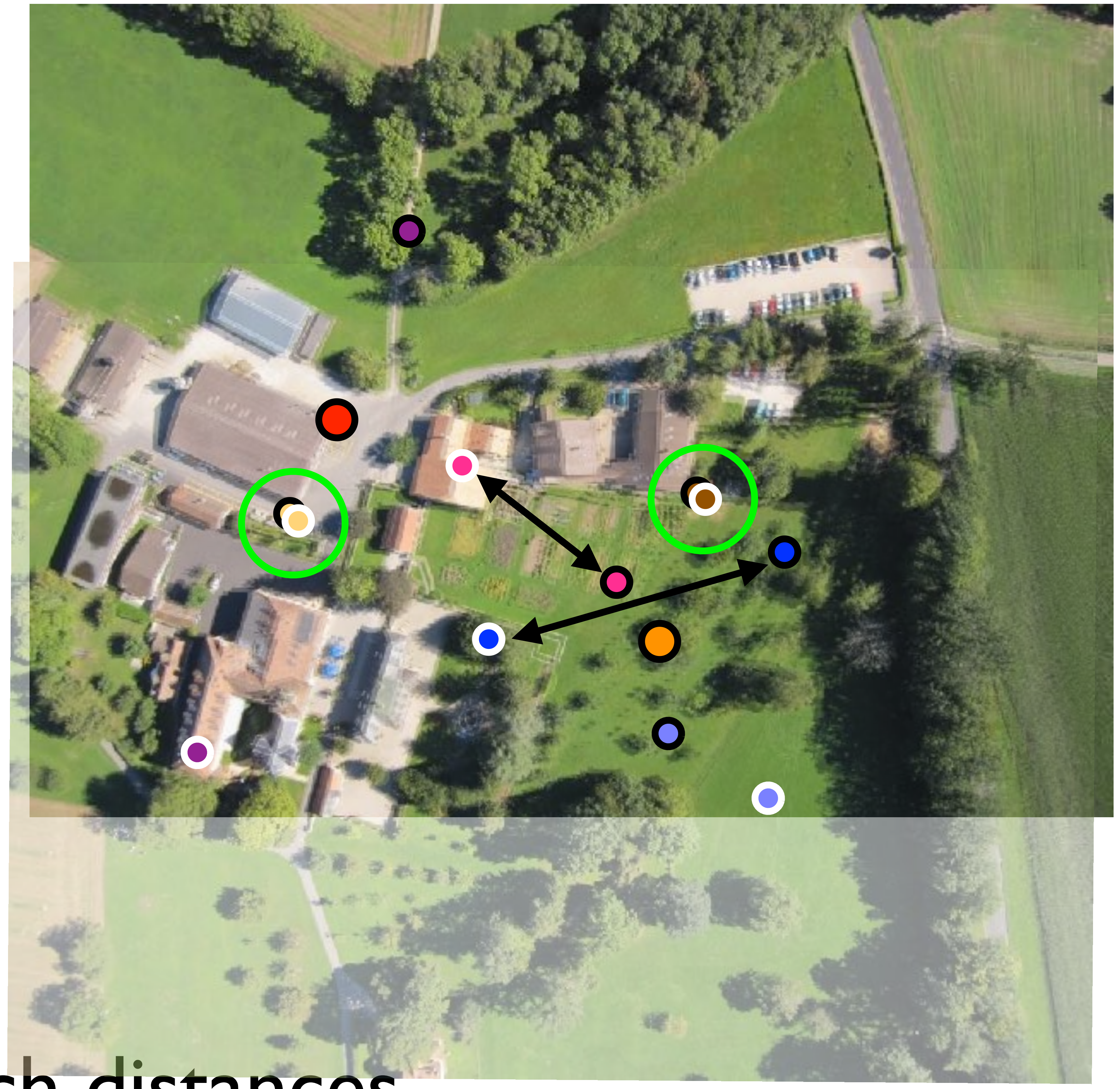
Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)

check match distances

Recap: Image **Alignment + RANSAC**

RANSAC solution for Similarity Transform (2 points)



check match distances

Recap: Image **Alignment + RANSAC**

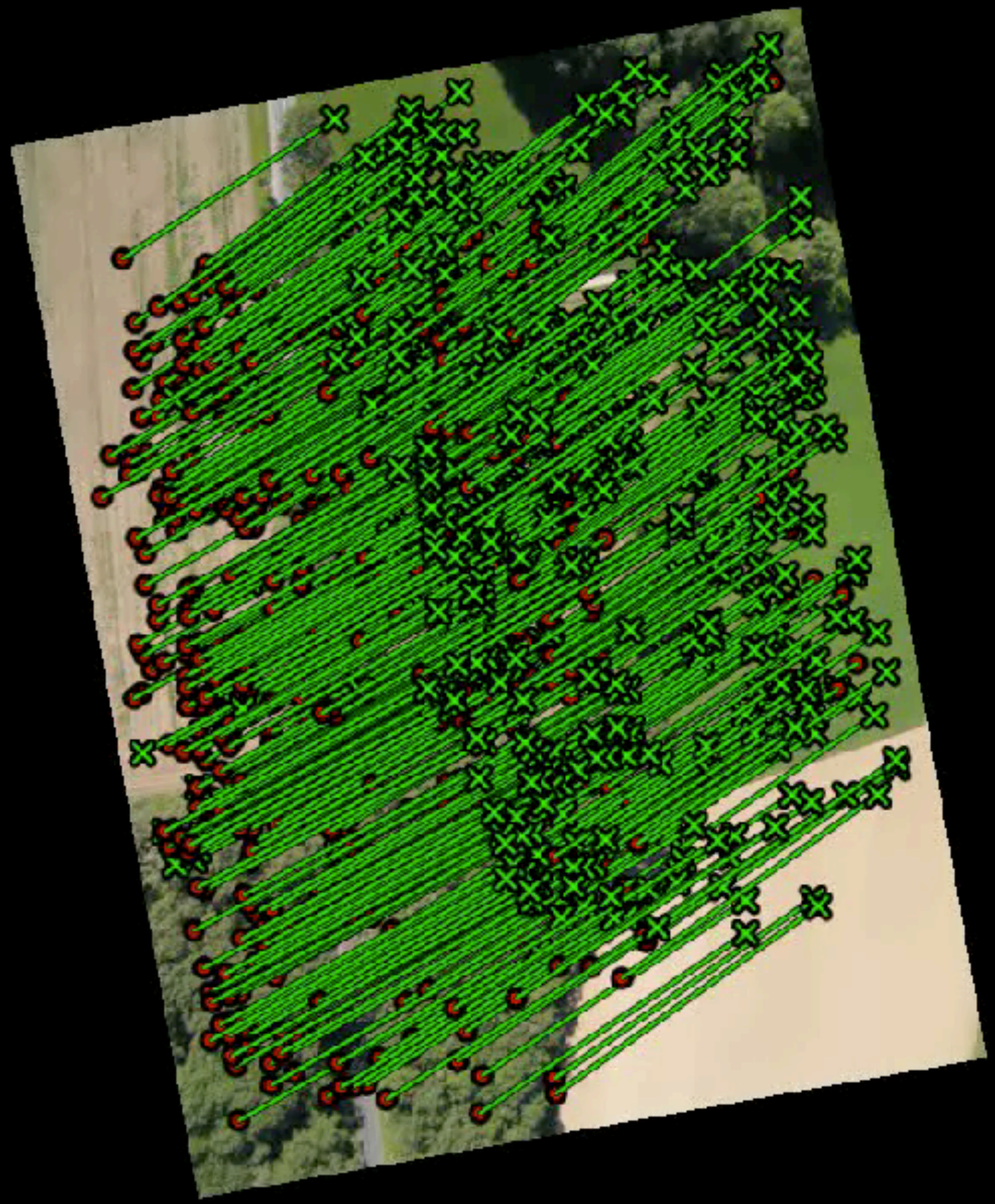
RANSAC solution for Similarity Transform (2 points)

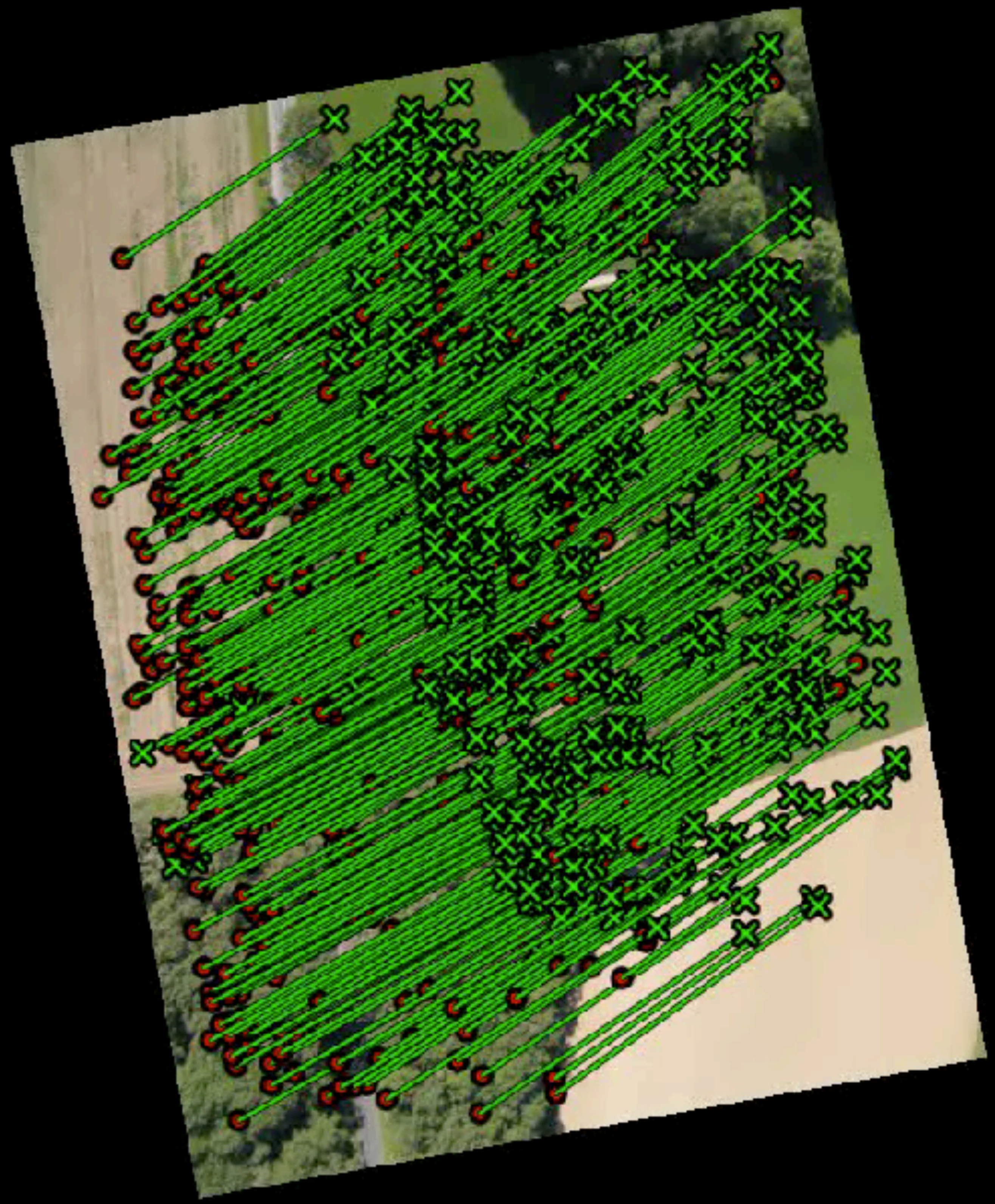
check match distances

#inliers = 4

Planar Image Alignment

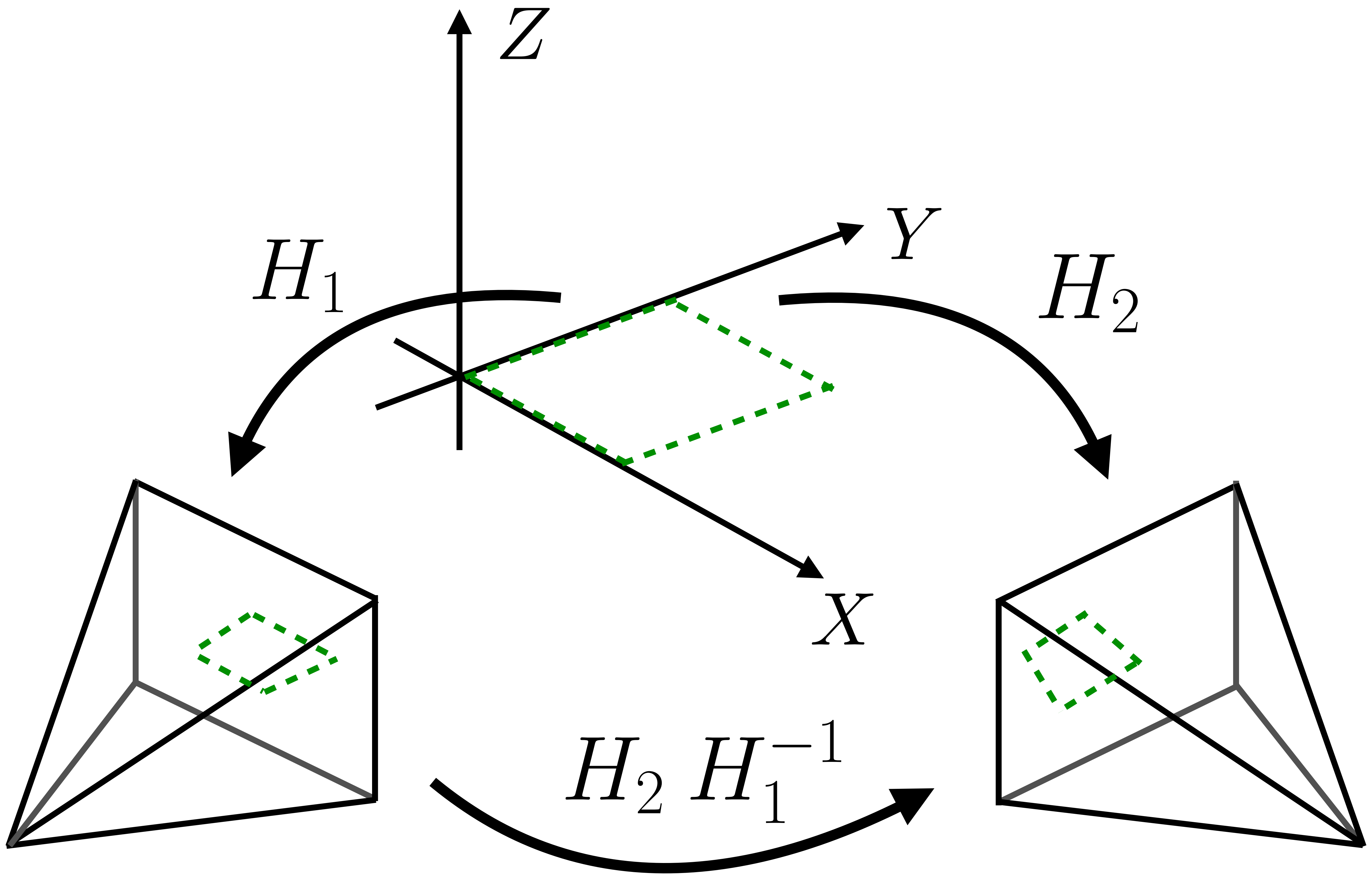
- Given a clean set of correspondences, align all images





Multiview Image Alignment

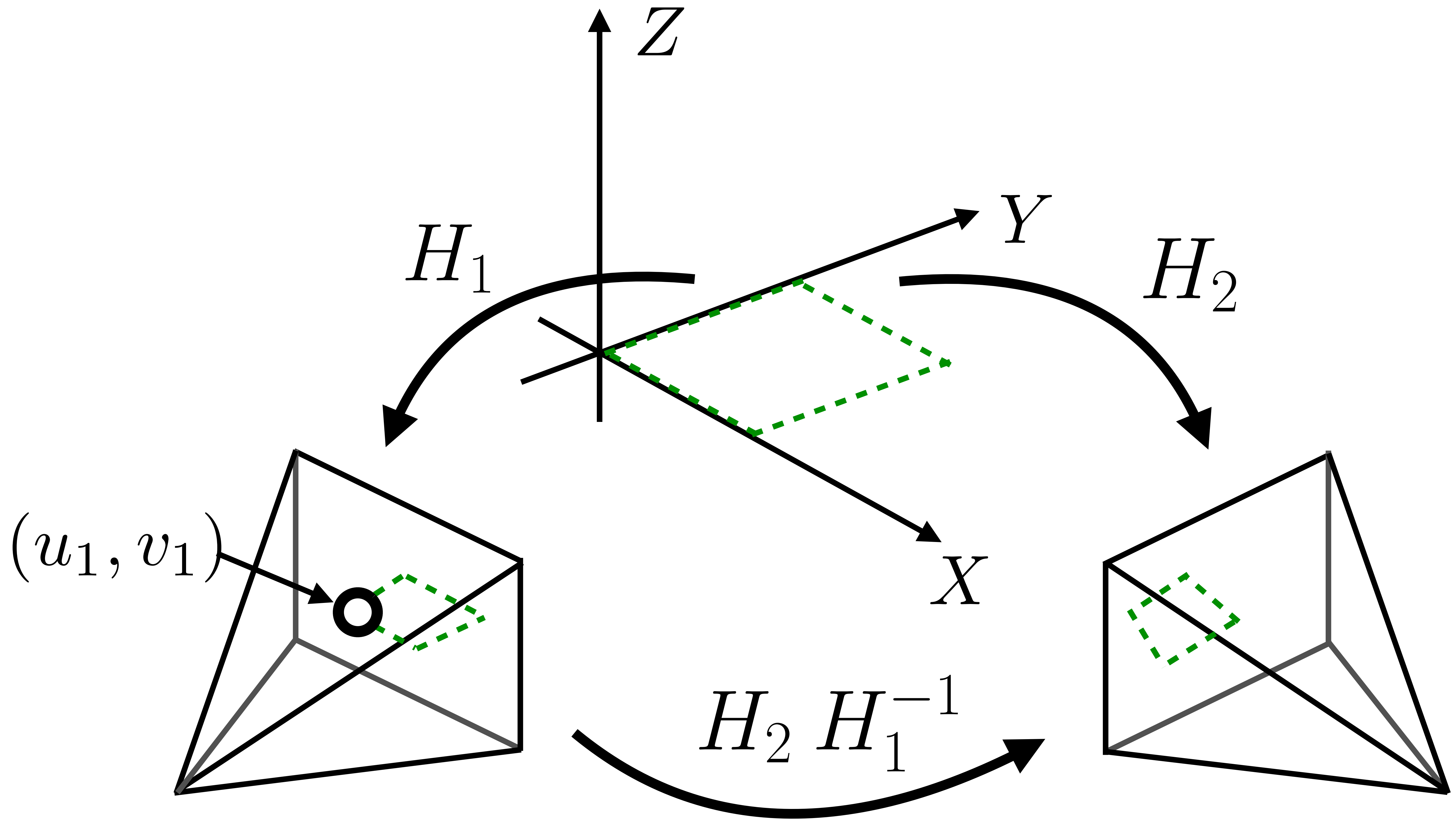
Residual = vector between observed feature and projection



16.1

Multiview Image Alignment

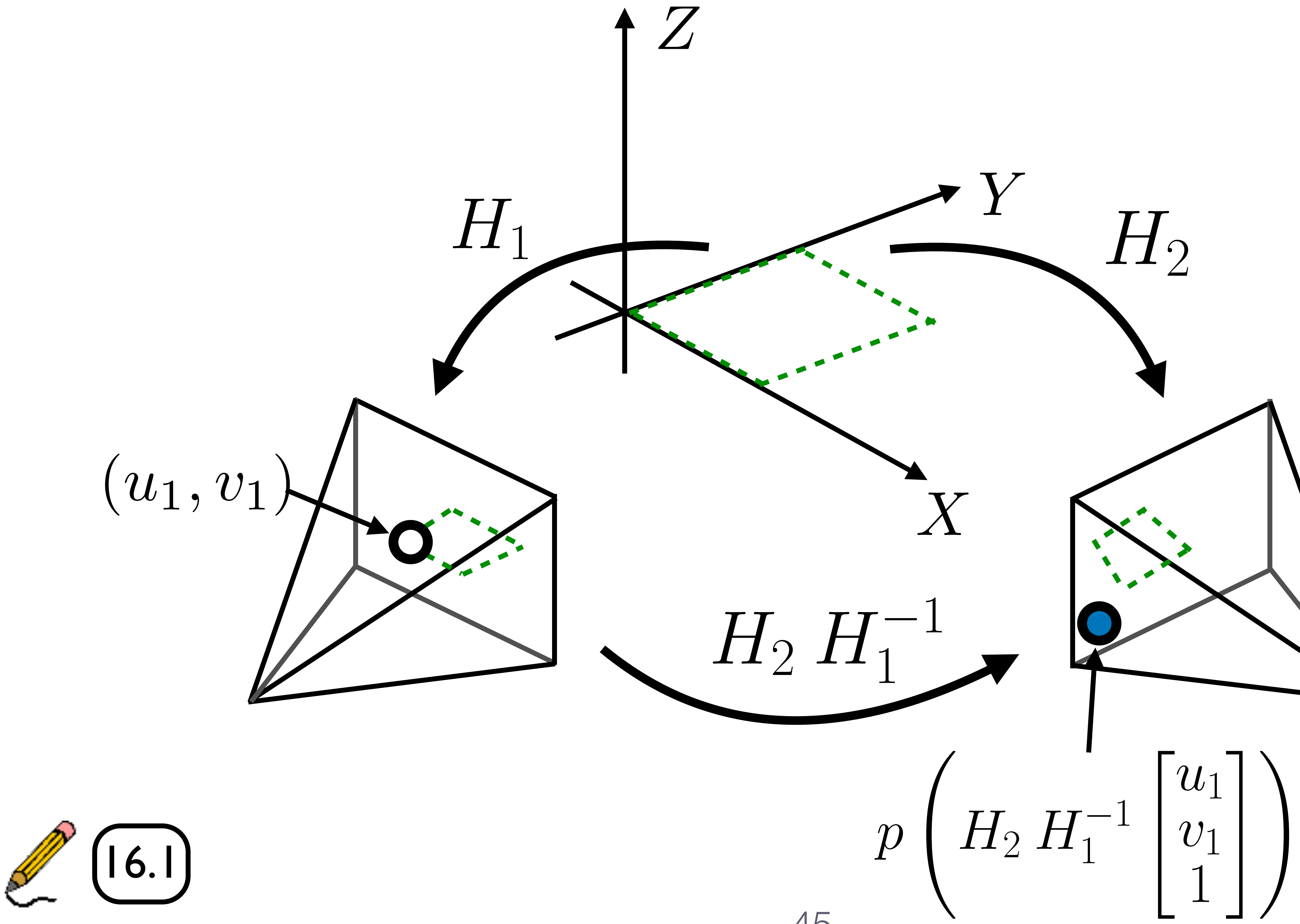
Residual = vector between observed feature and projection



16.1

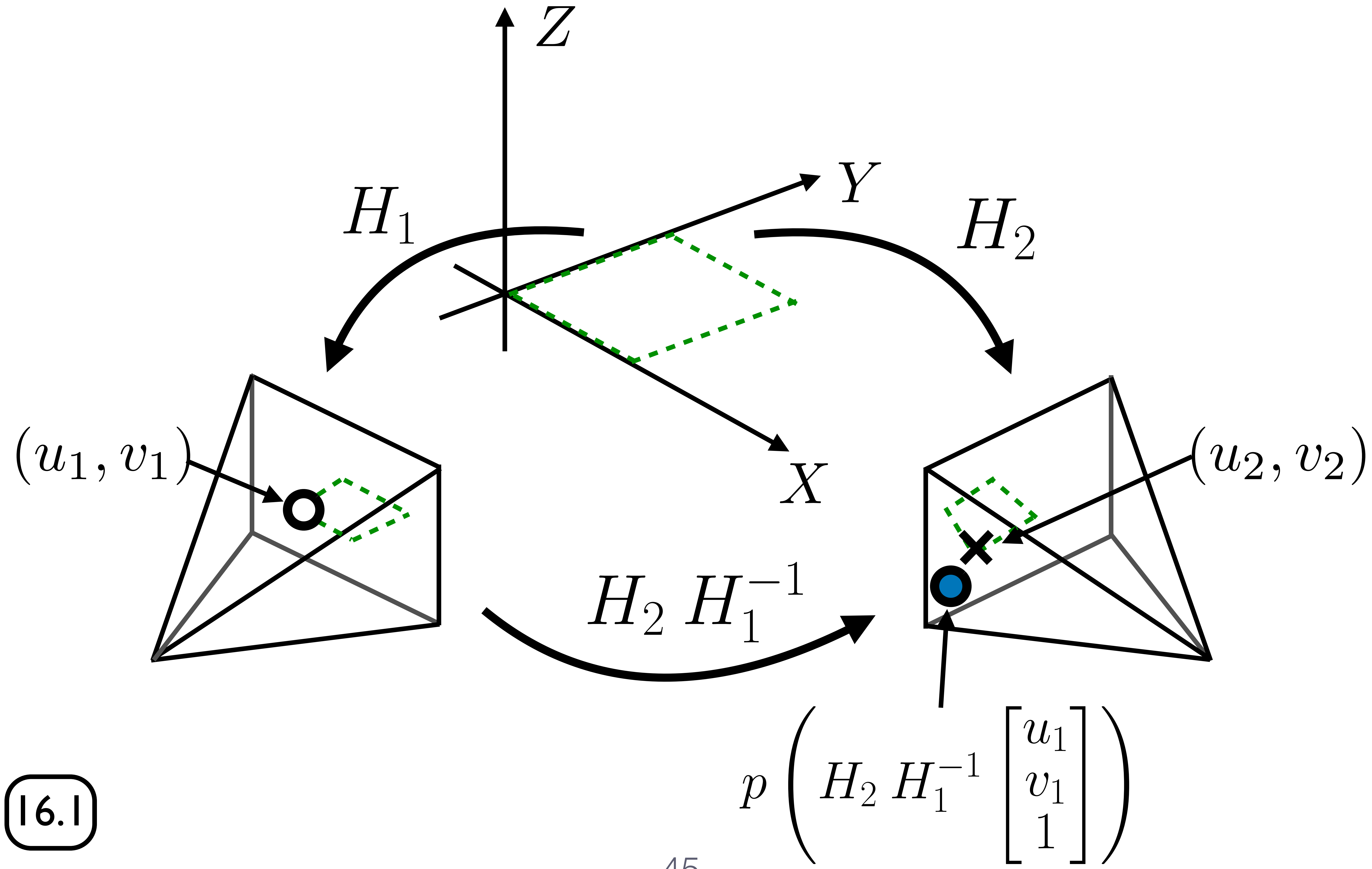
Multiview Image Alignment

Residual = vector between observed feature and projection



Multiview Image Alignment

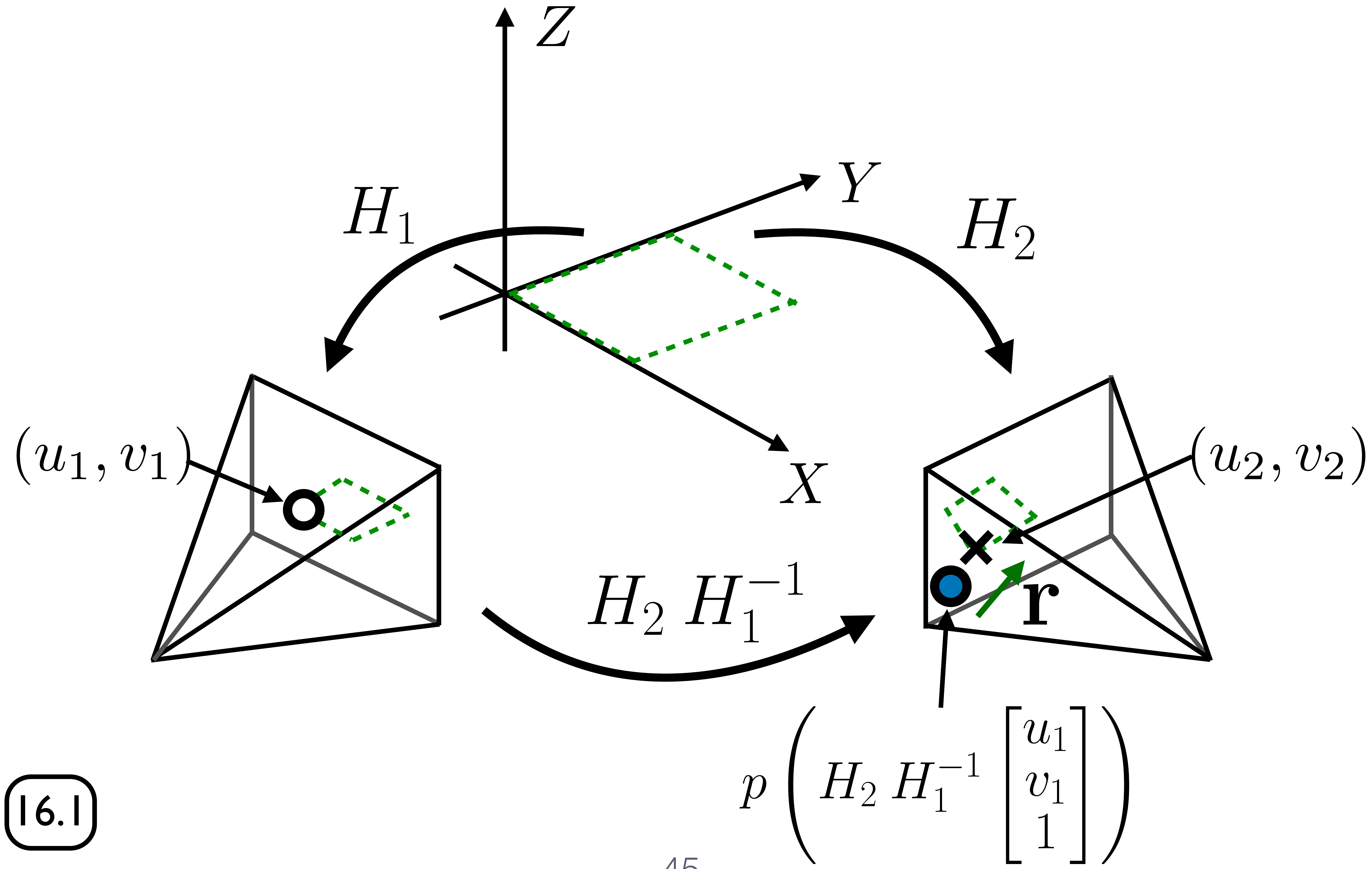
Residual = vector between observed feature and projection



16.1

Multiview Image Alignment

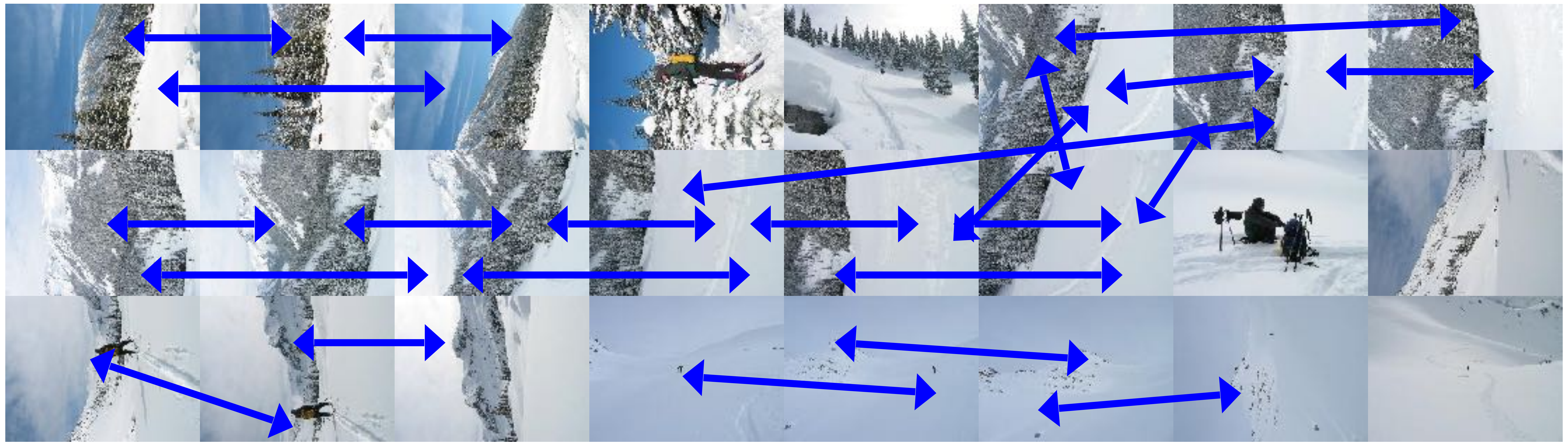
Residual = vector between observed feature and projection



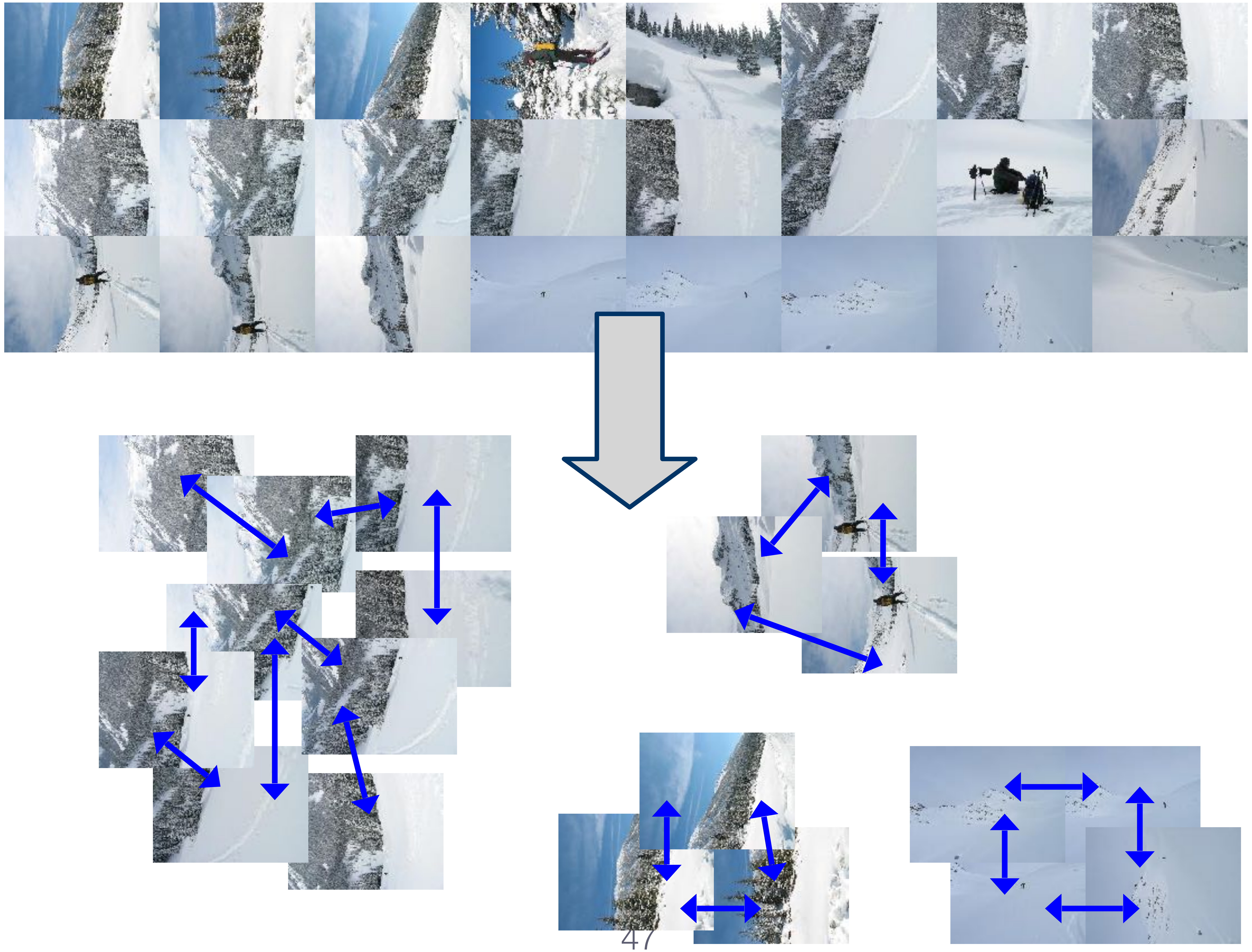
16.1

Panorama Recognition

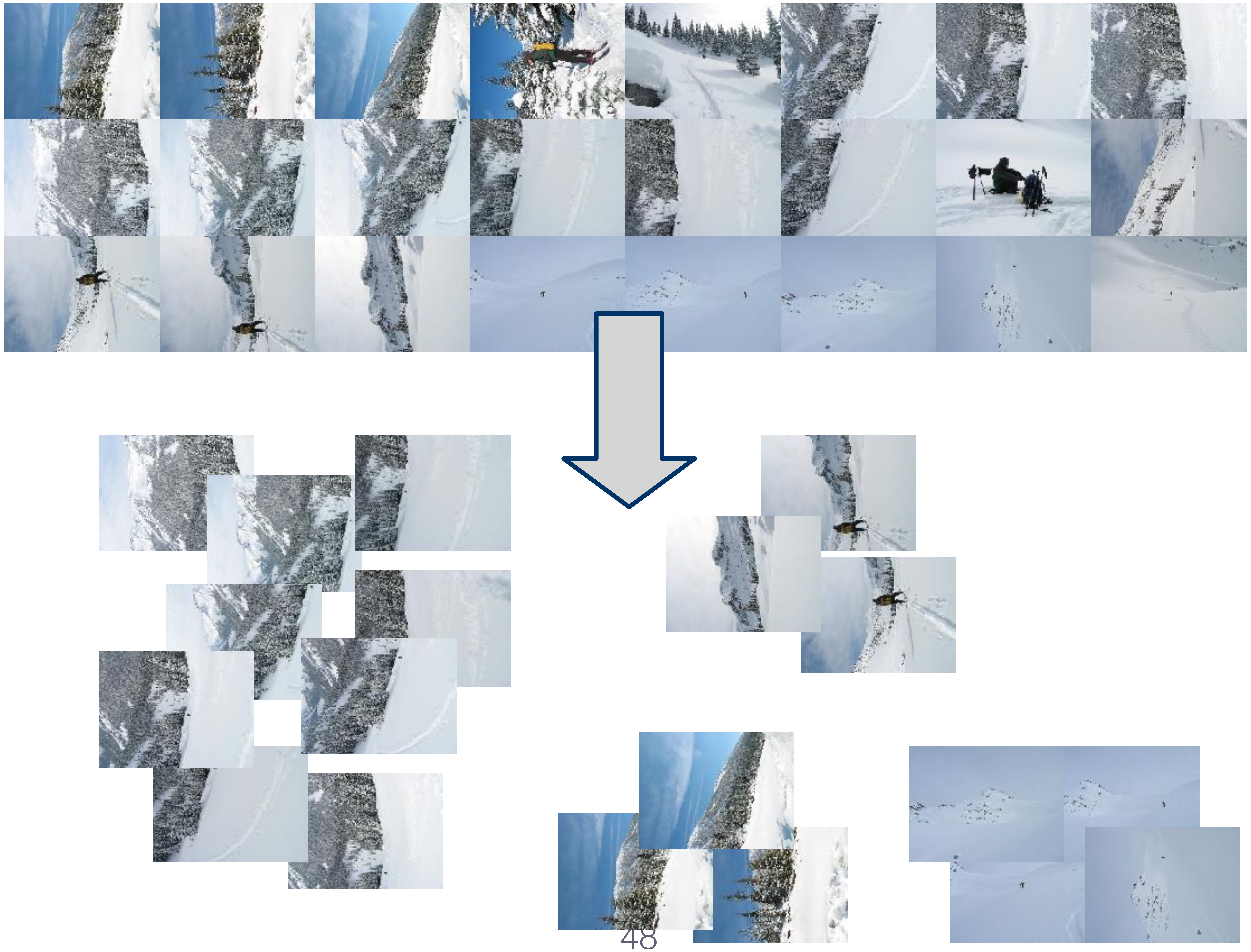
Panorama Recognition



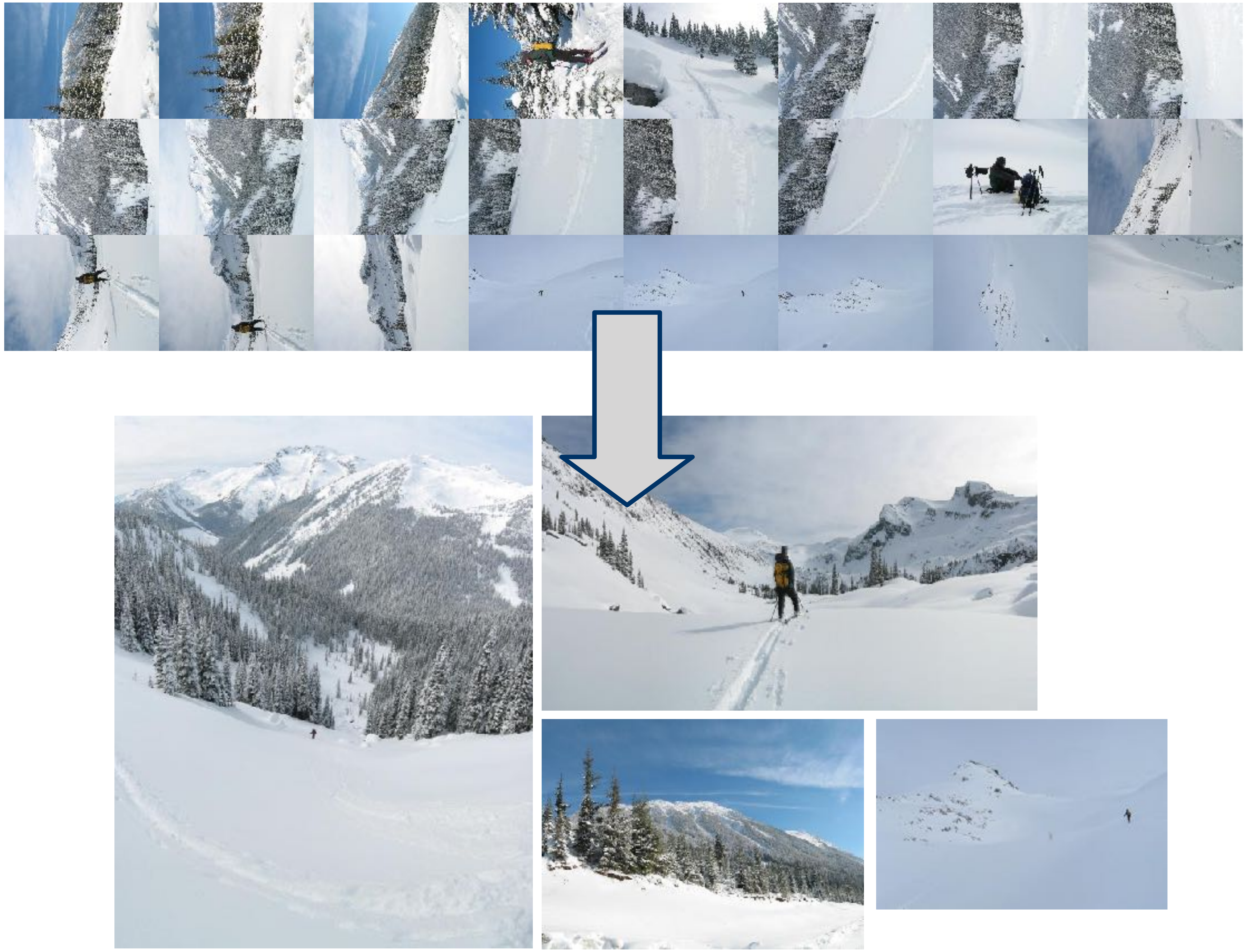
Panorama Recognition



Panorama Recognition



Panorama Recognition



Building a panorama

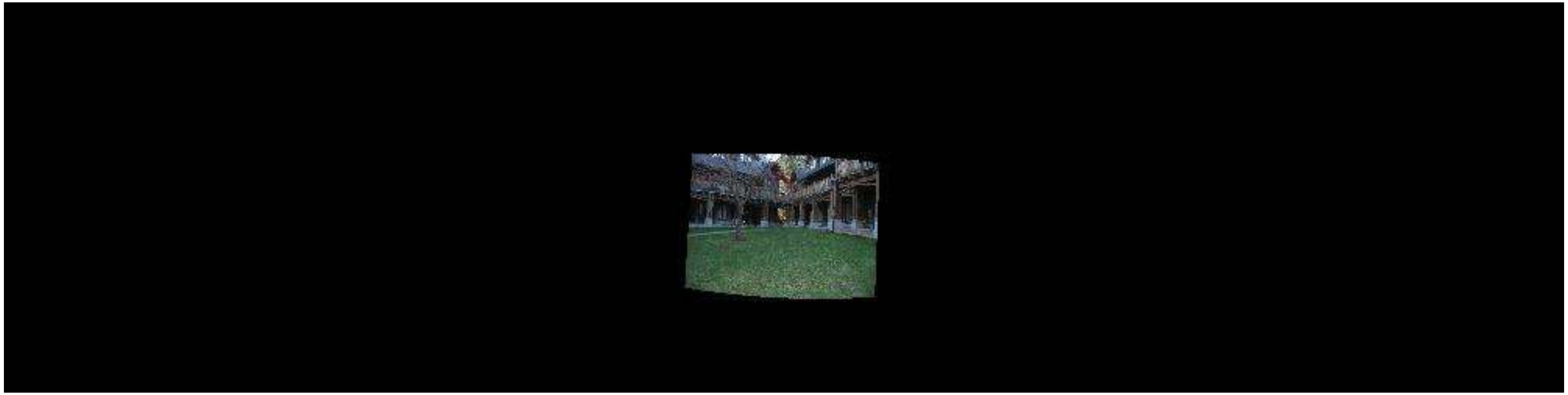


Figure Credit: Matthew Brown and David Lowe

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

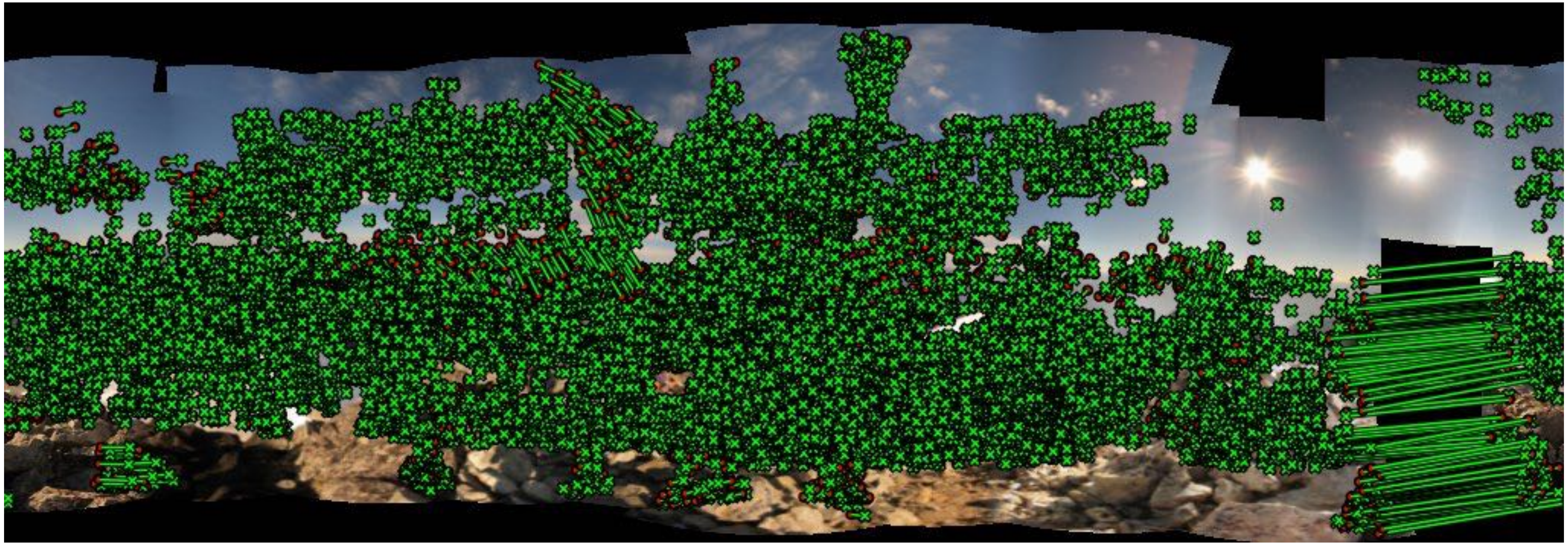
Figure Credit: Matthew Brown and David Lowe

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

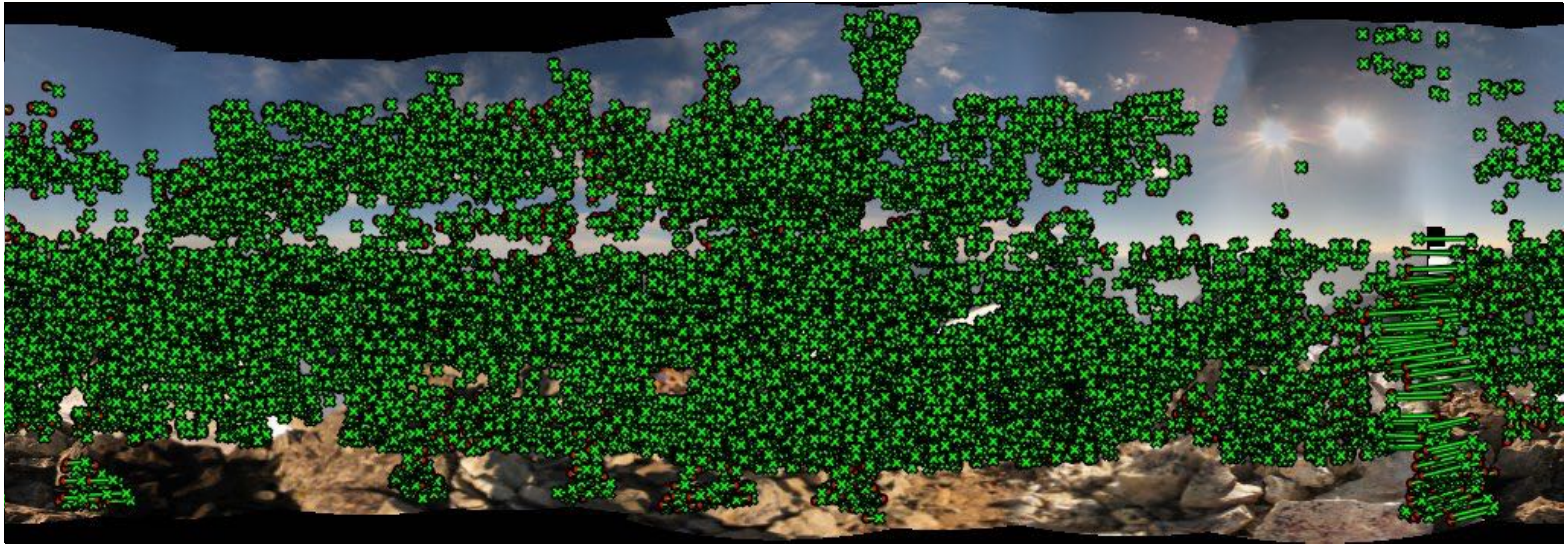
Panorama Stitching

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



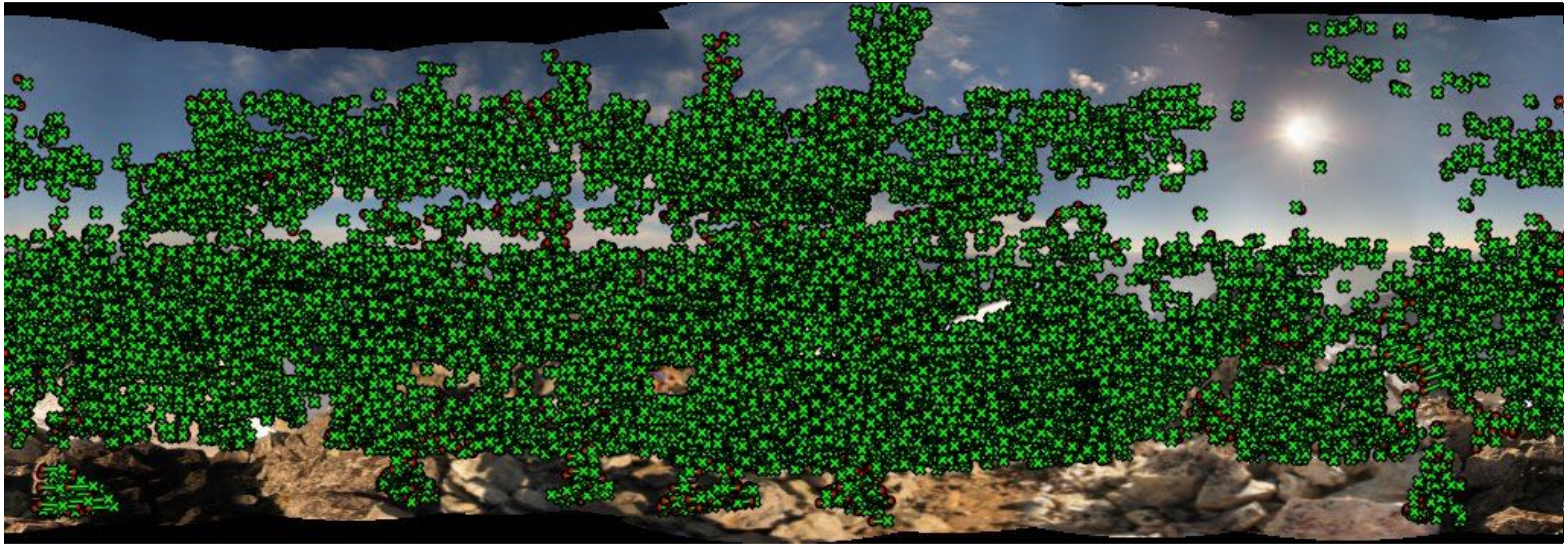
Panorama Stitching

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



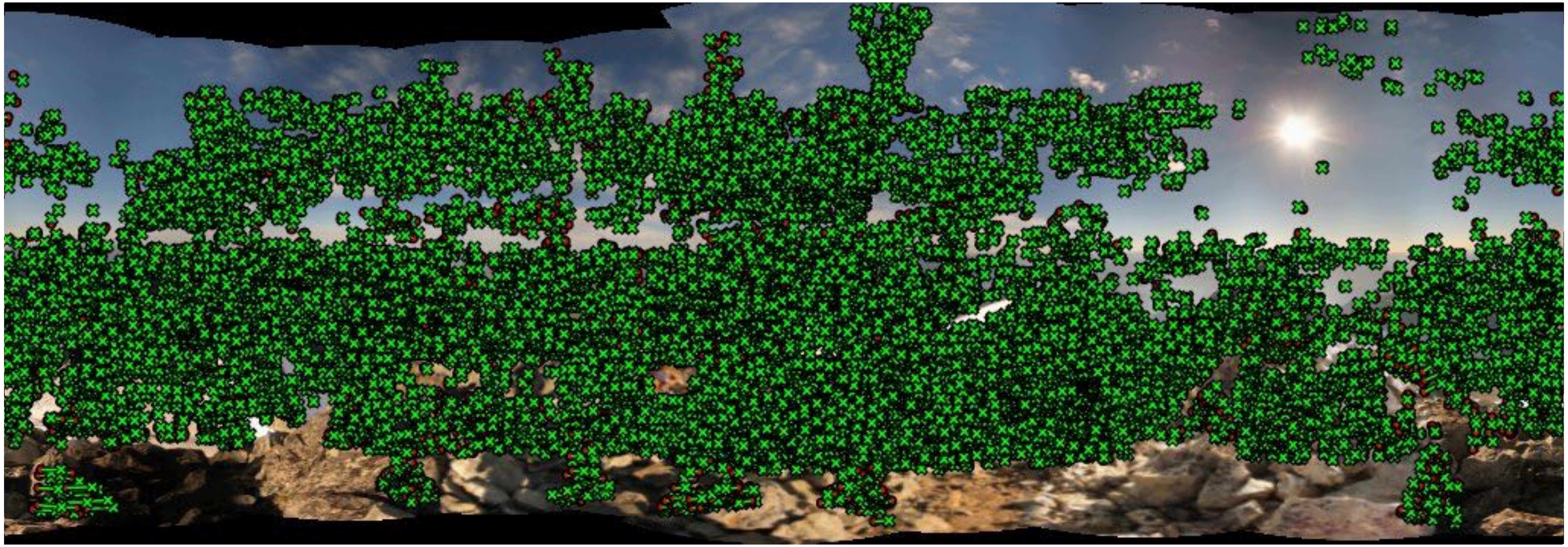
Panorama Stitching

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



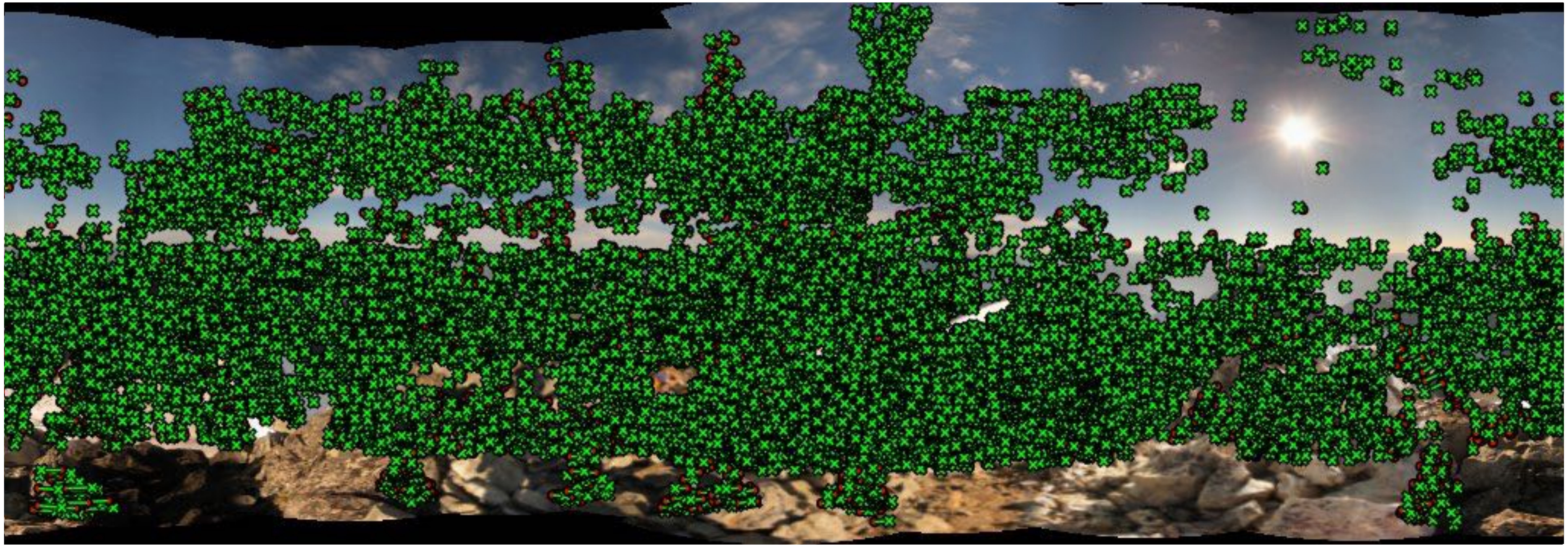
Panorama Stitching

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



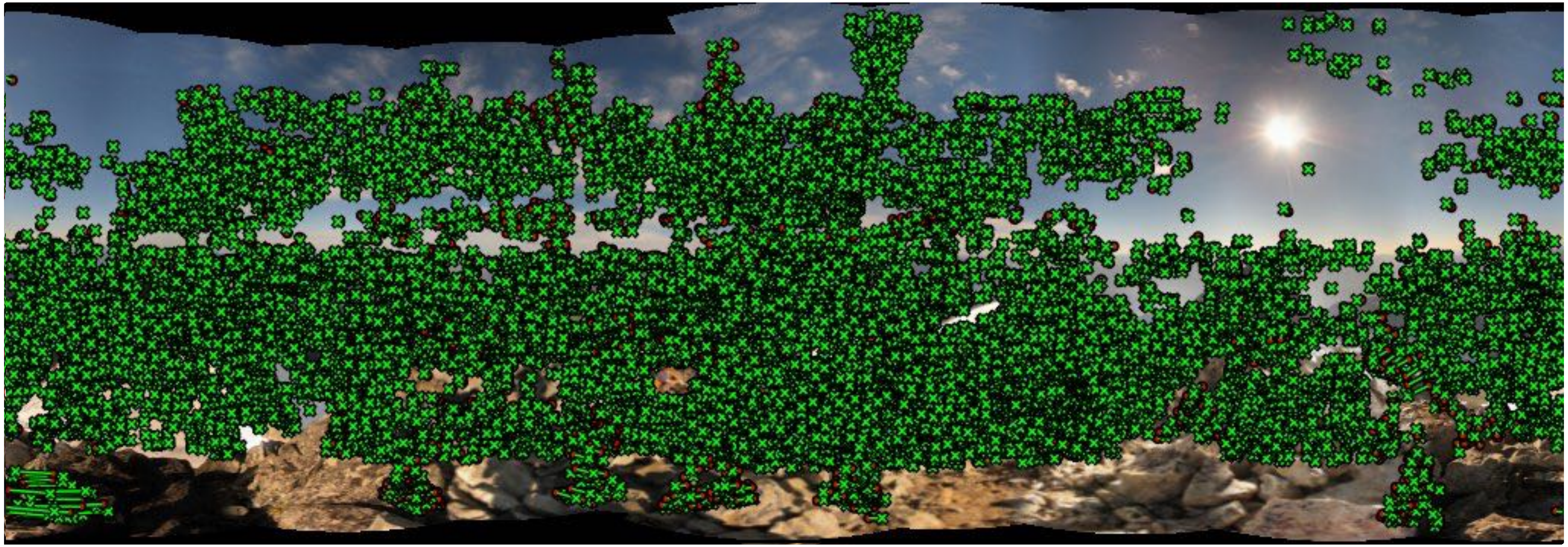
Panorama Stitching

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



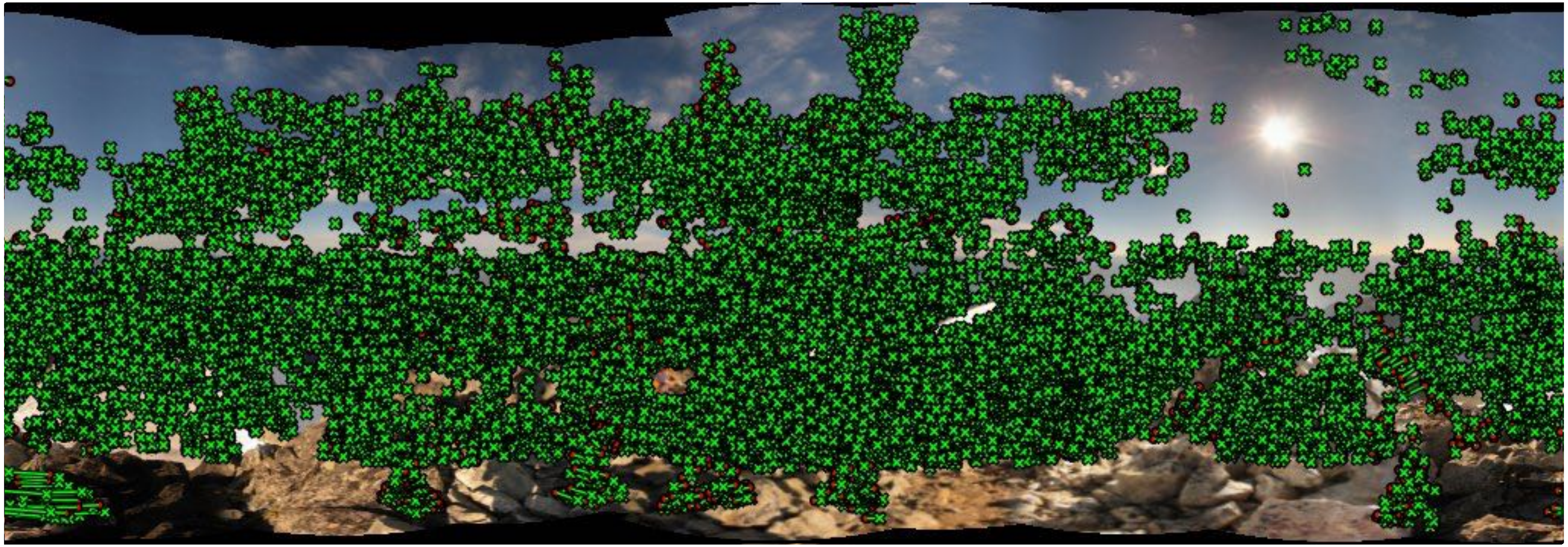
Panorama Stitching

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



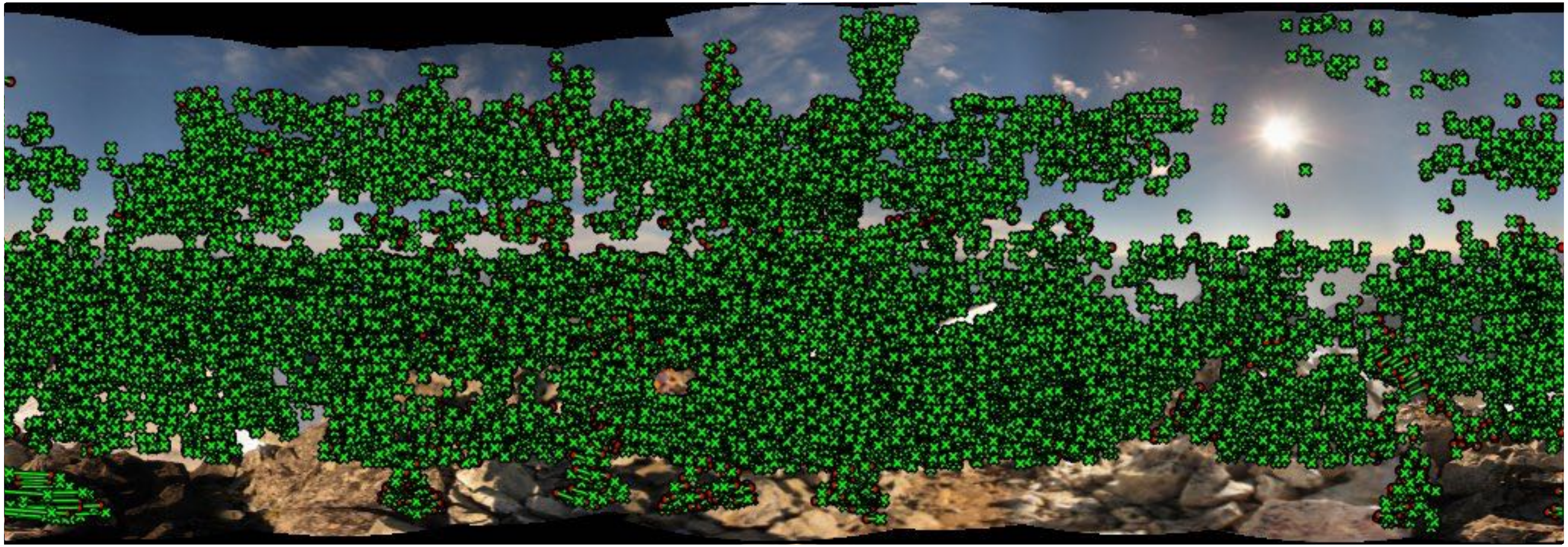
Panorama Stitching

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



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



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

Structure from Motion

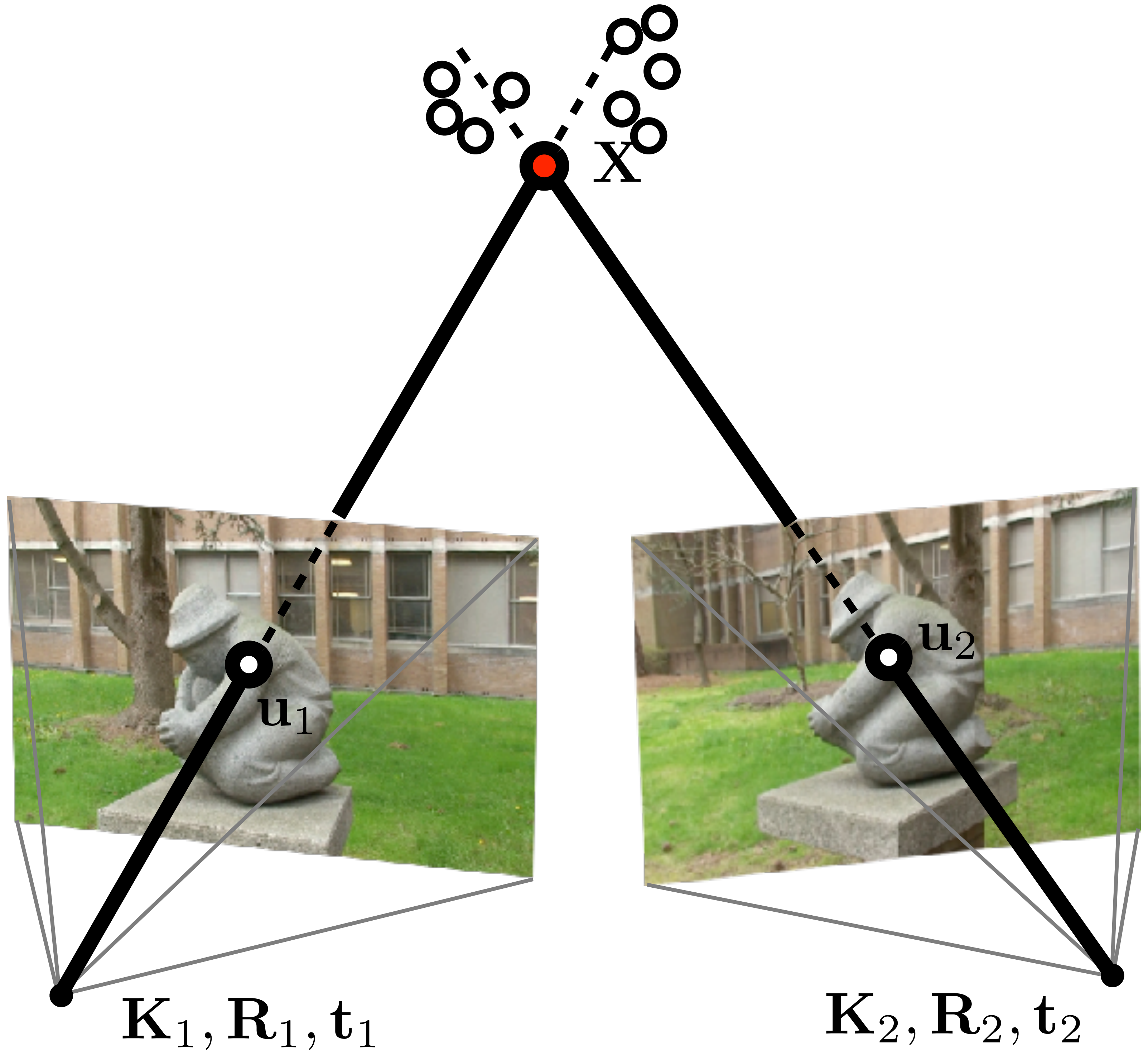
2-view Structure from Motion

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

Raw SIFT matches

RANSAC for Epipolar Geom

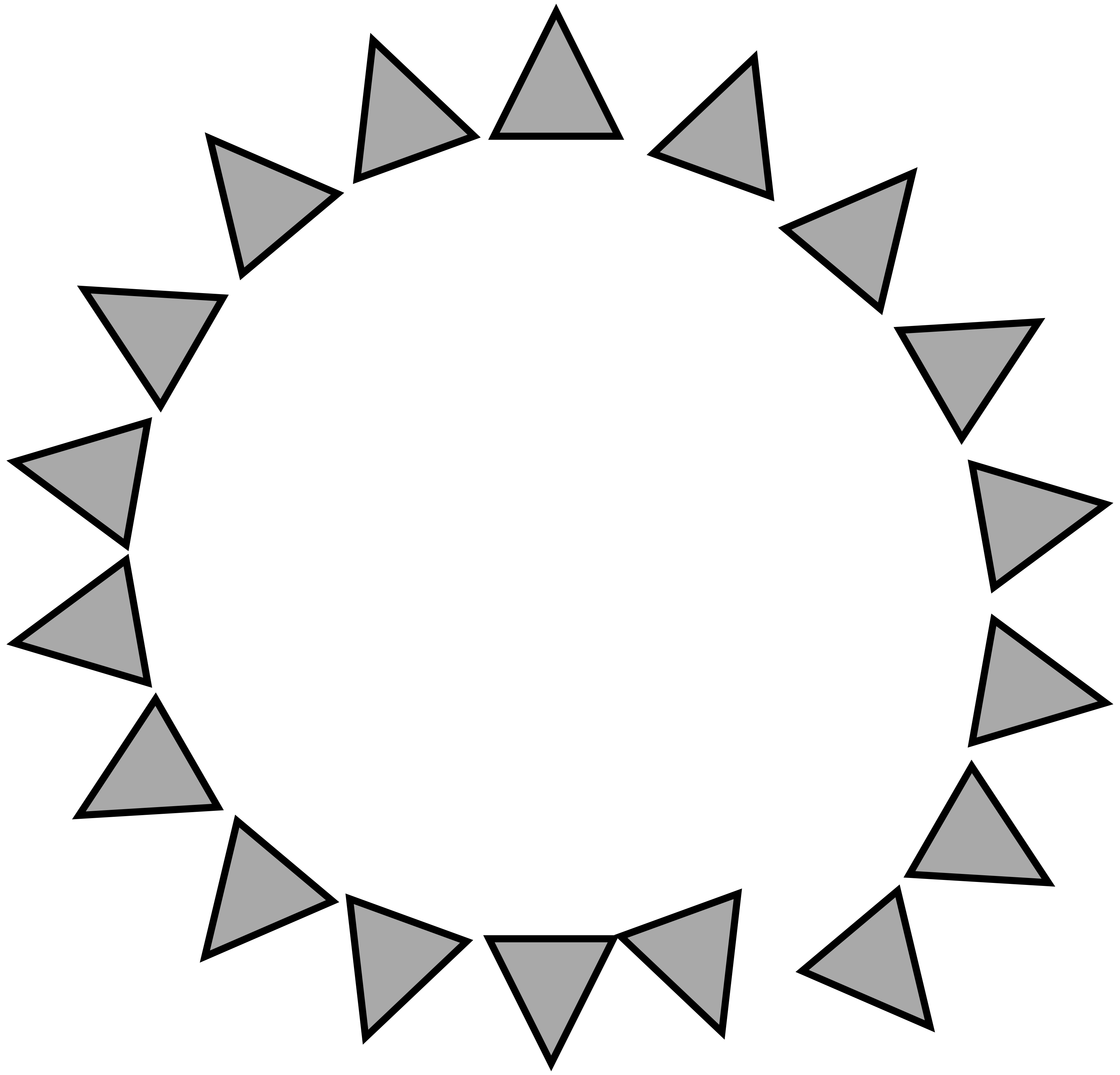
Extract R, t



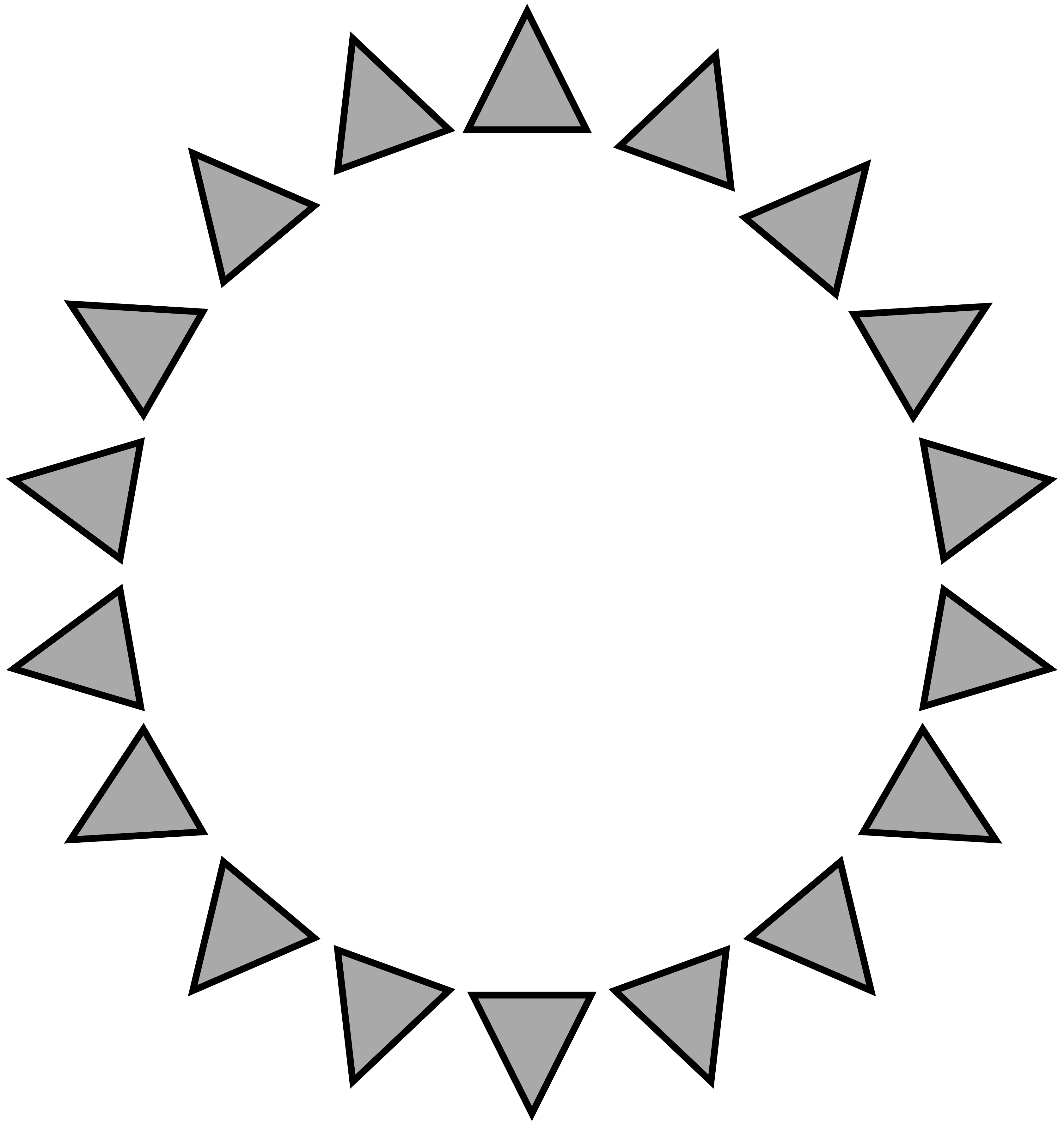
Triangulate to 3D Point Cloud

Global Alignment

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



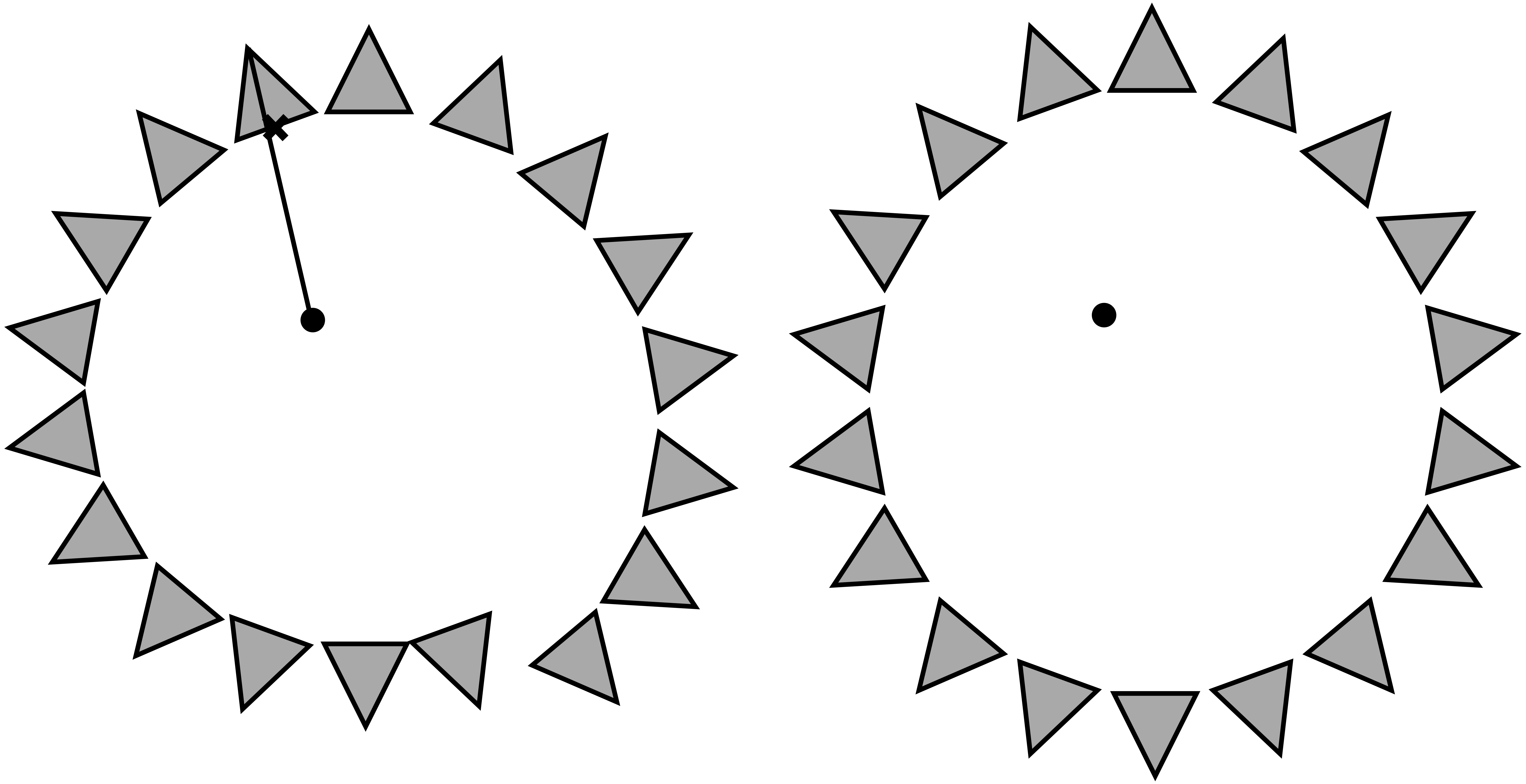
Pairwise alignment



Global alignment

Global Alignment

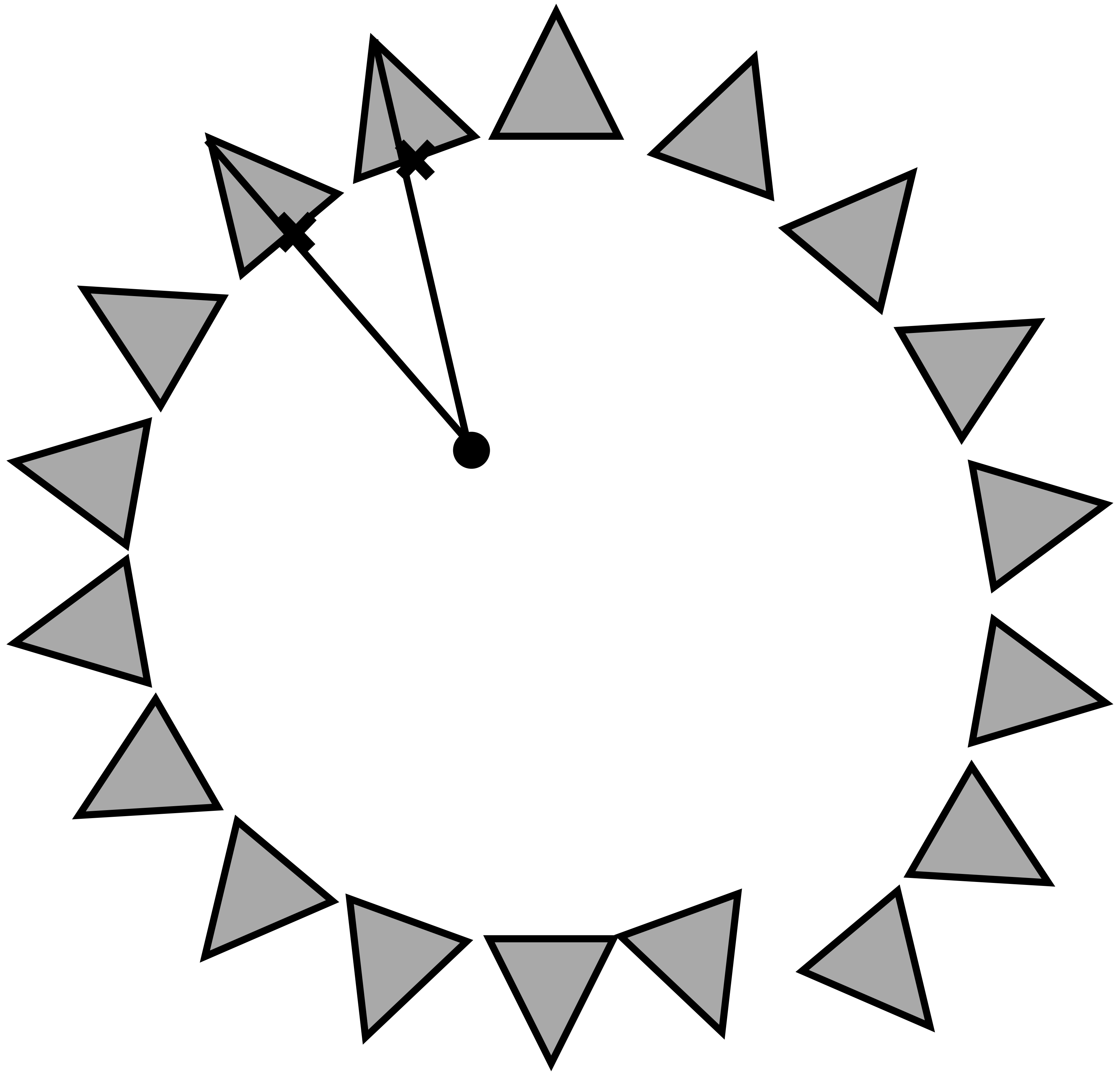
- Concatenation of pairwise R, t estimates results in drift, e.g.,



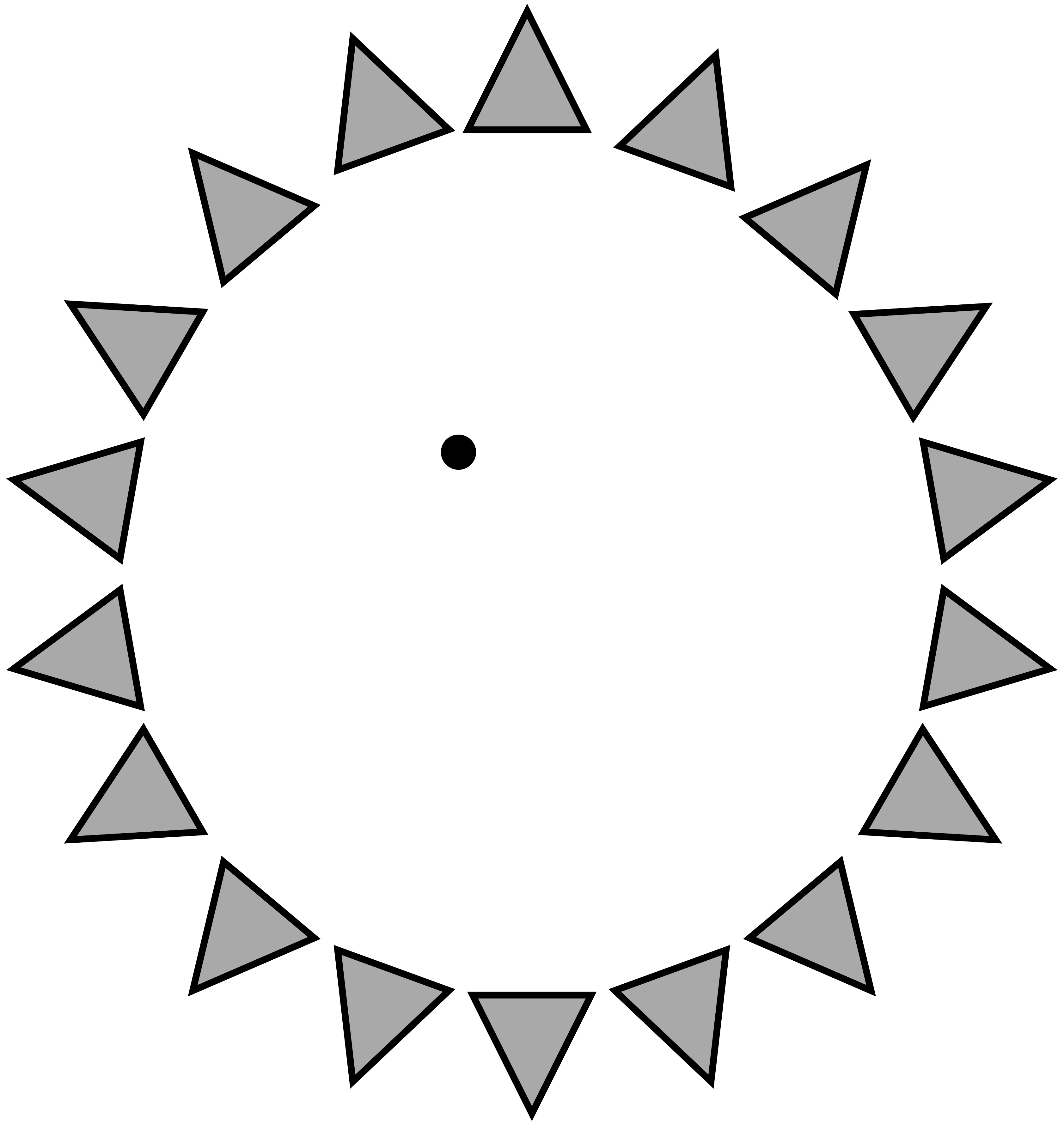
Pairwise alignment　　　　Global alignment

Global Alignment

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



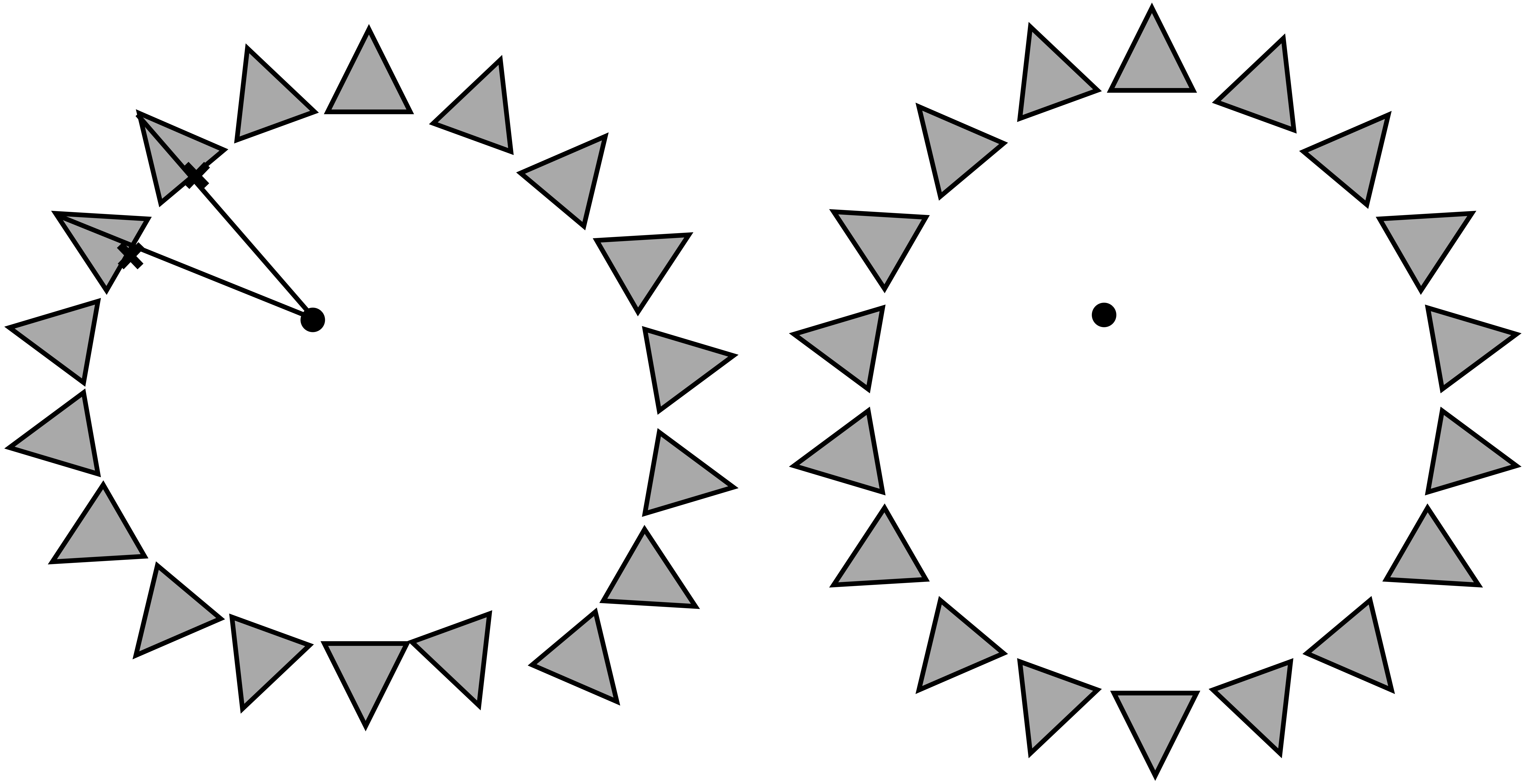
Pairwise alignment



Global alignment

Global Alignment

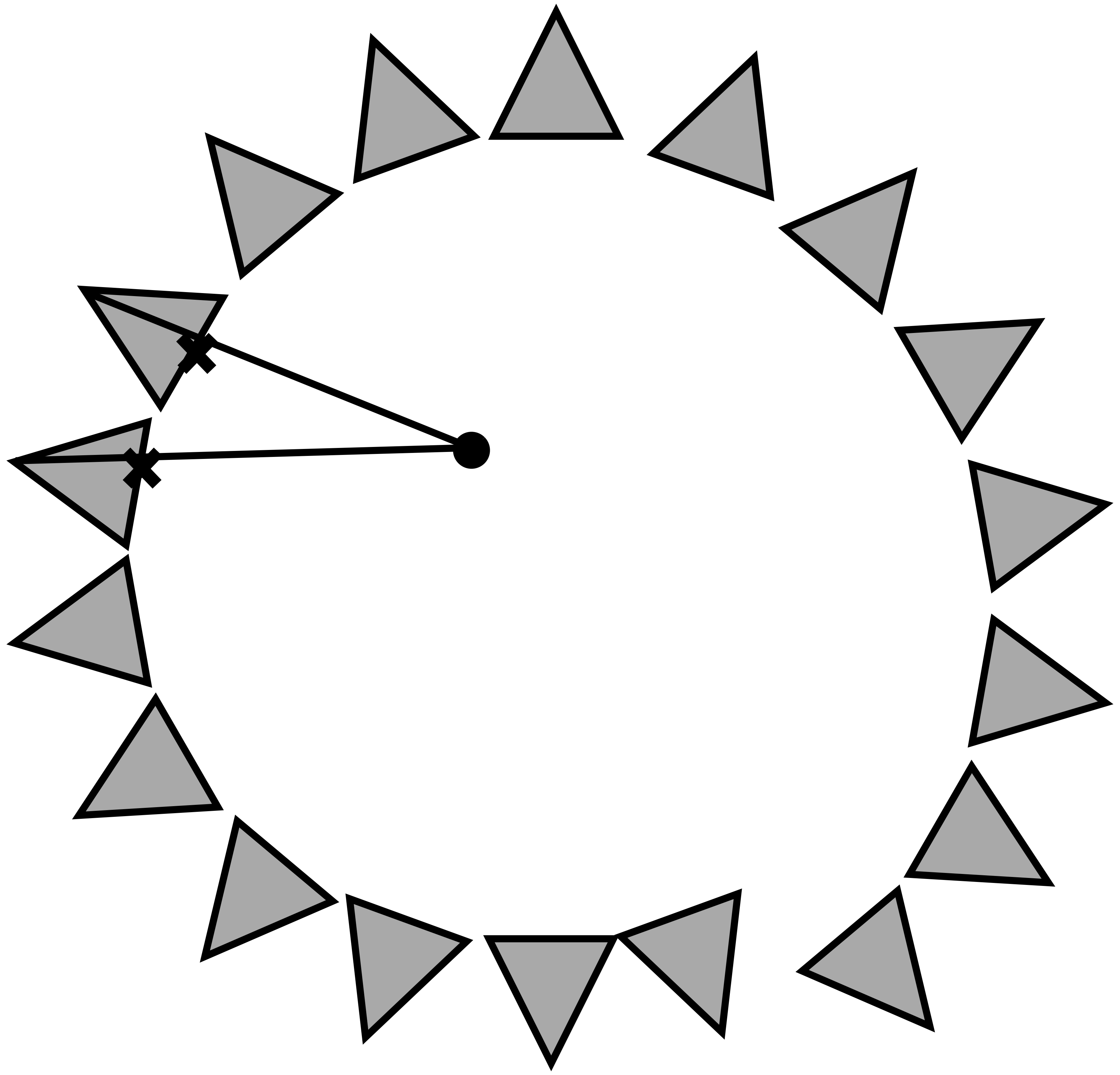
- Concatenation of pairwise R, t estimates results in drift, e.g.,



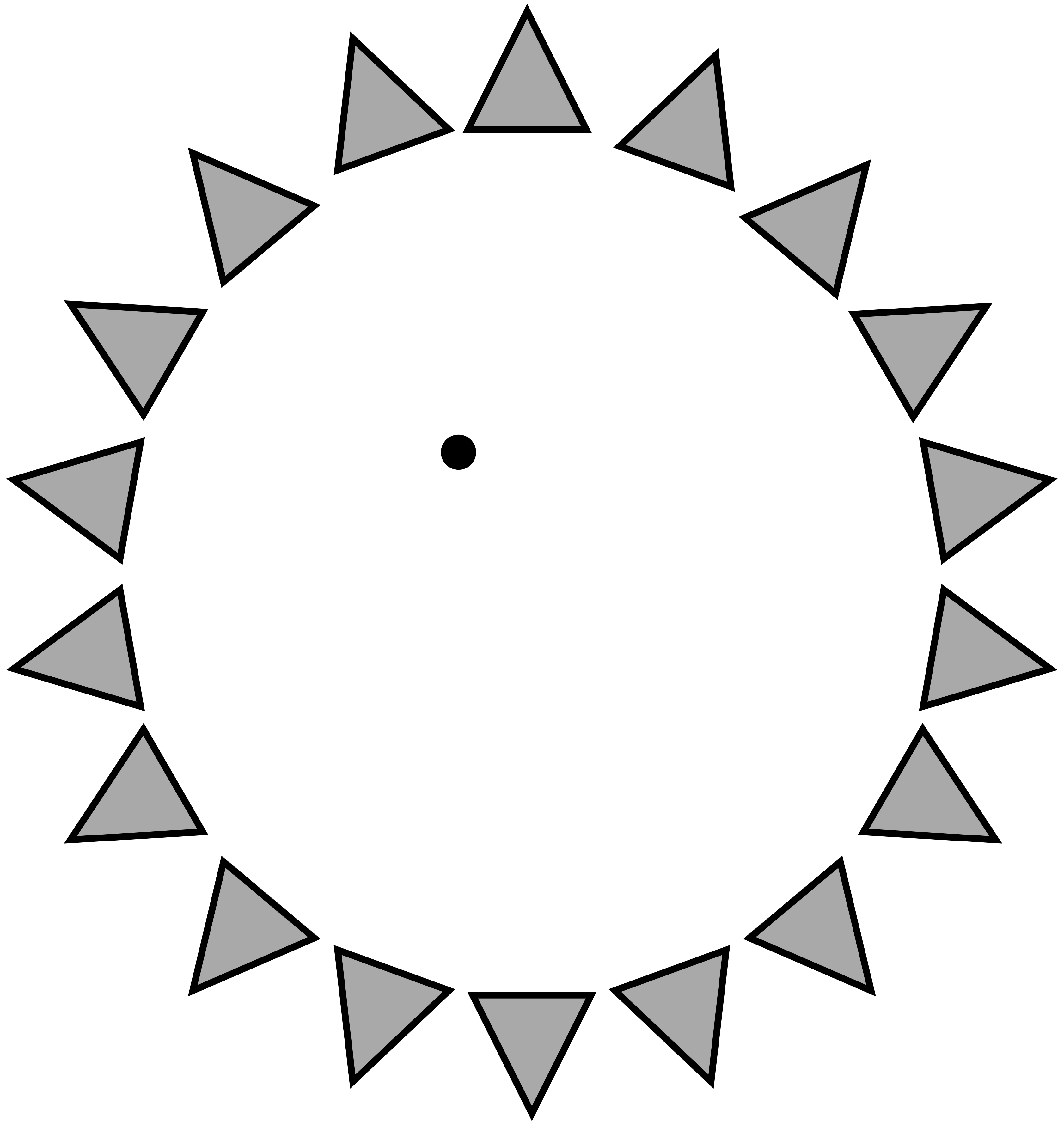
Pairwise alignment   Global alignment

Global Alignment

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



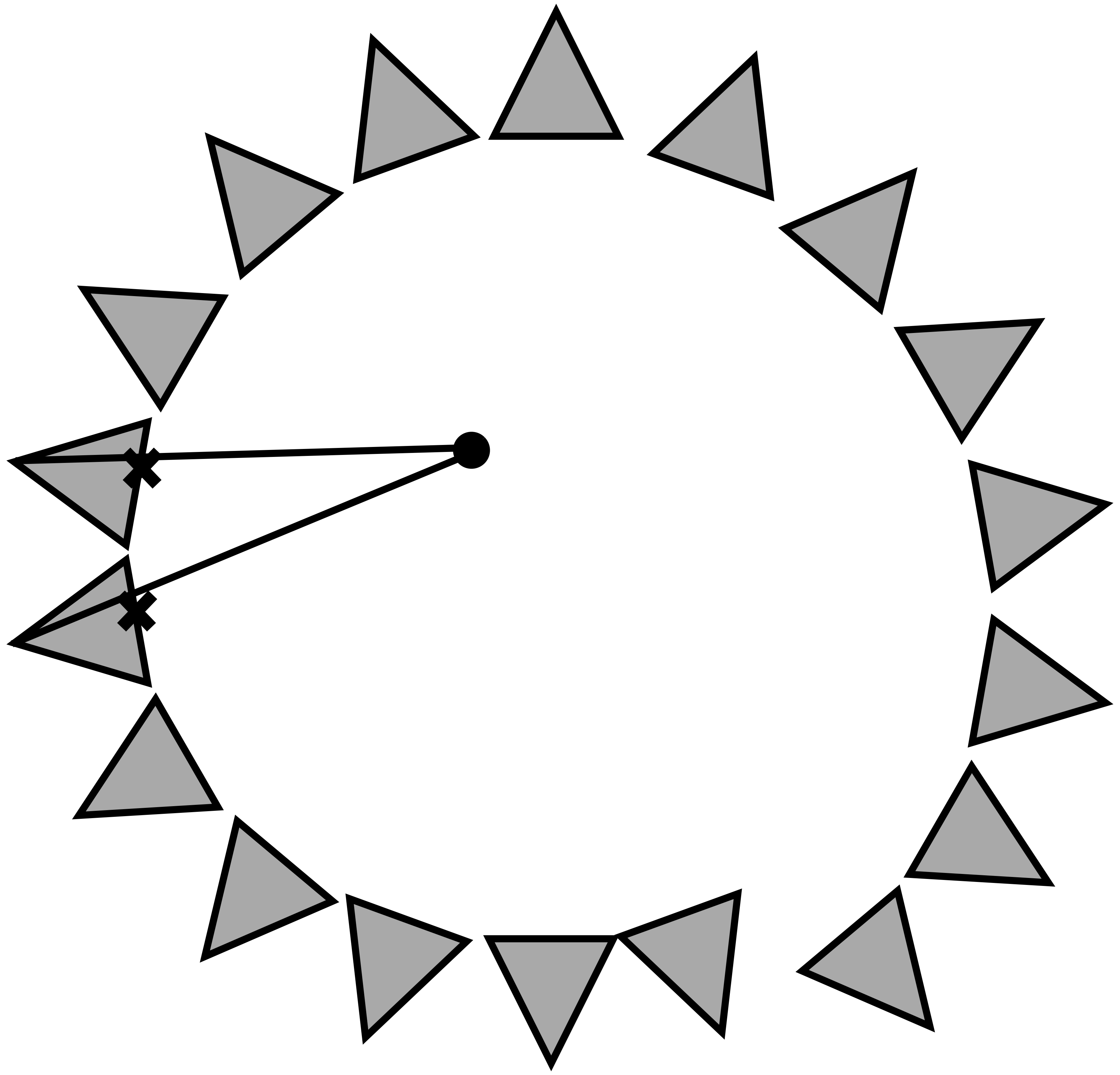
Pairwise alignment



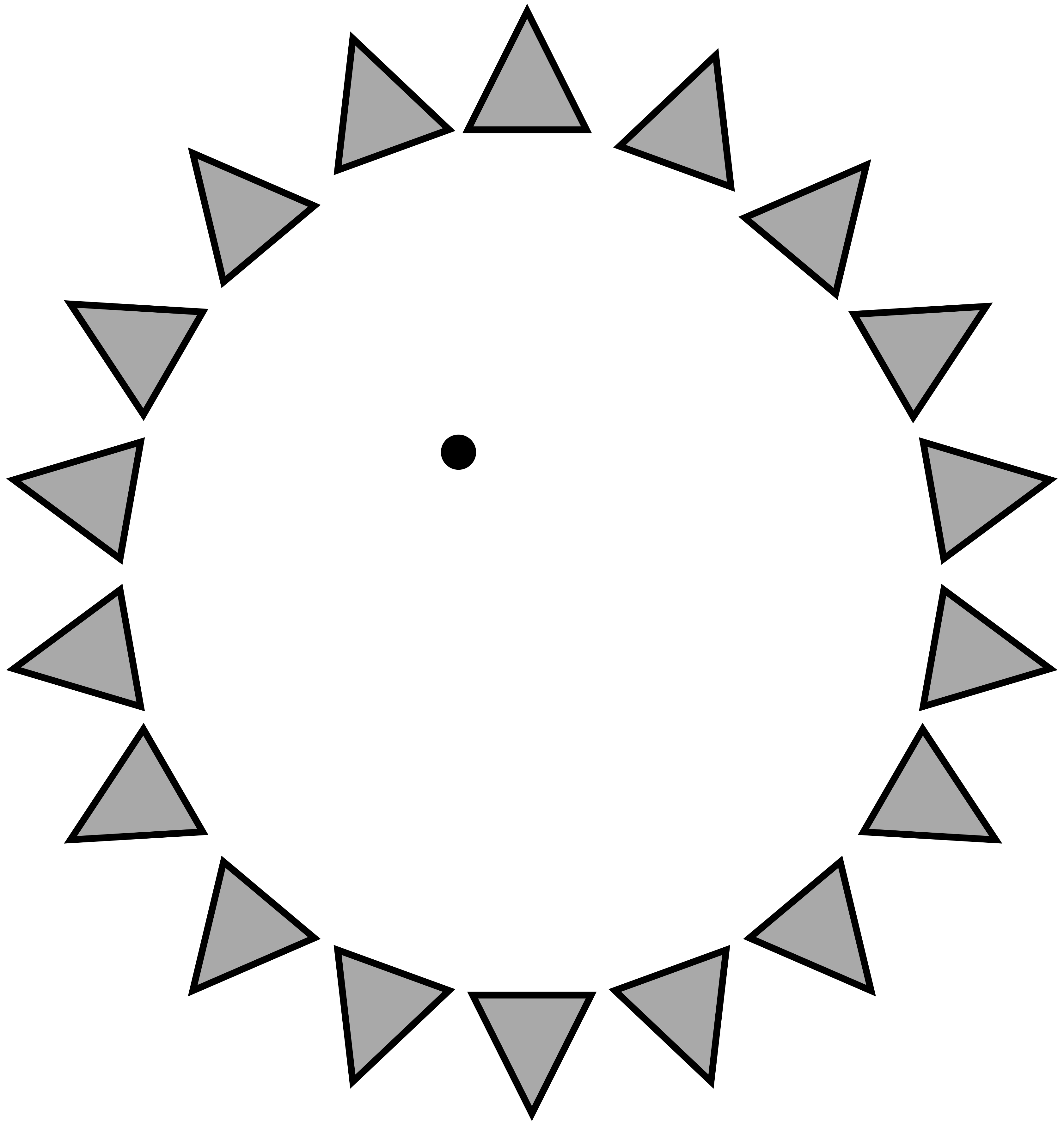
Global alignment

Global Alignment

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



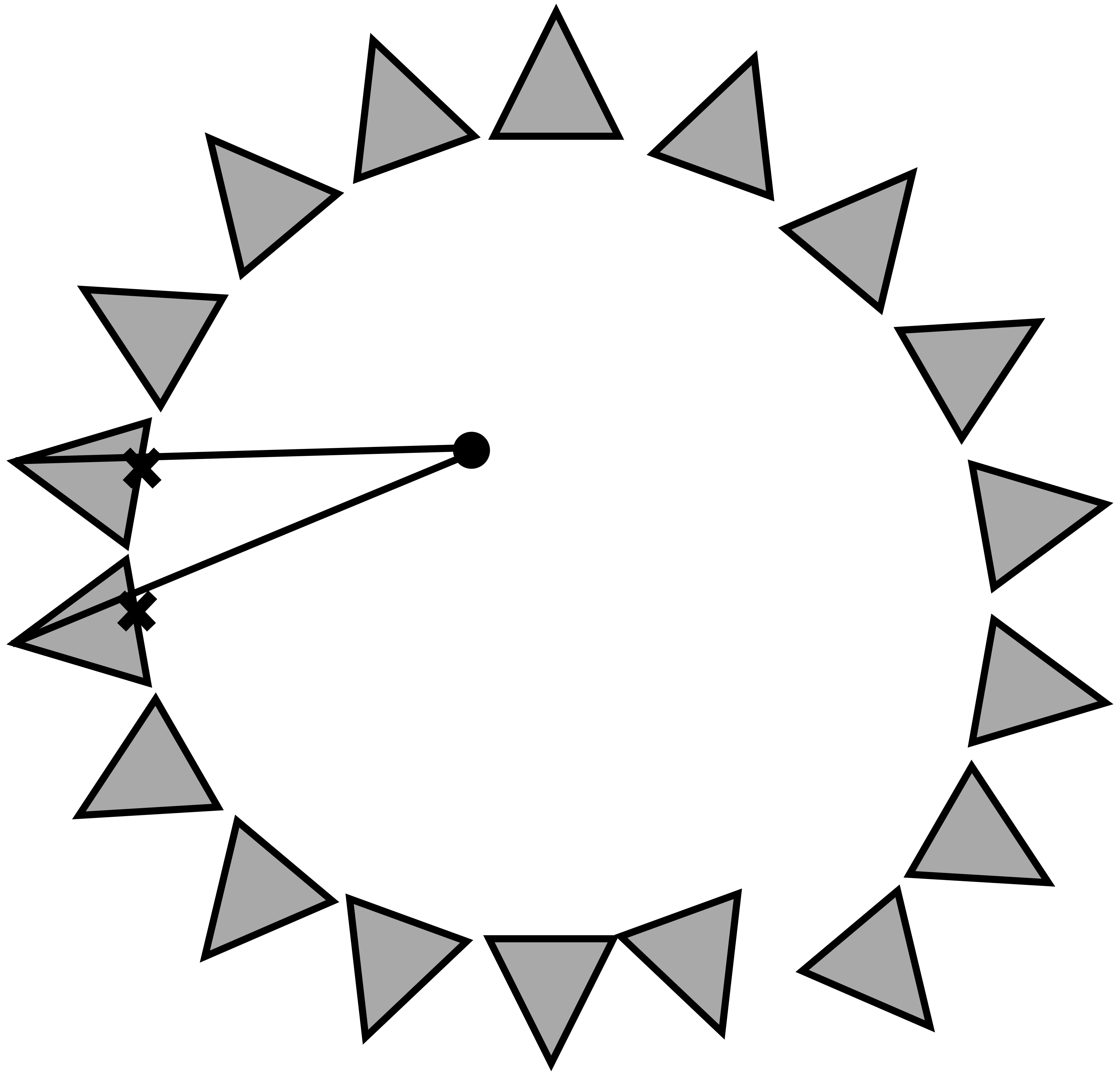
Pairwise alignment

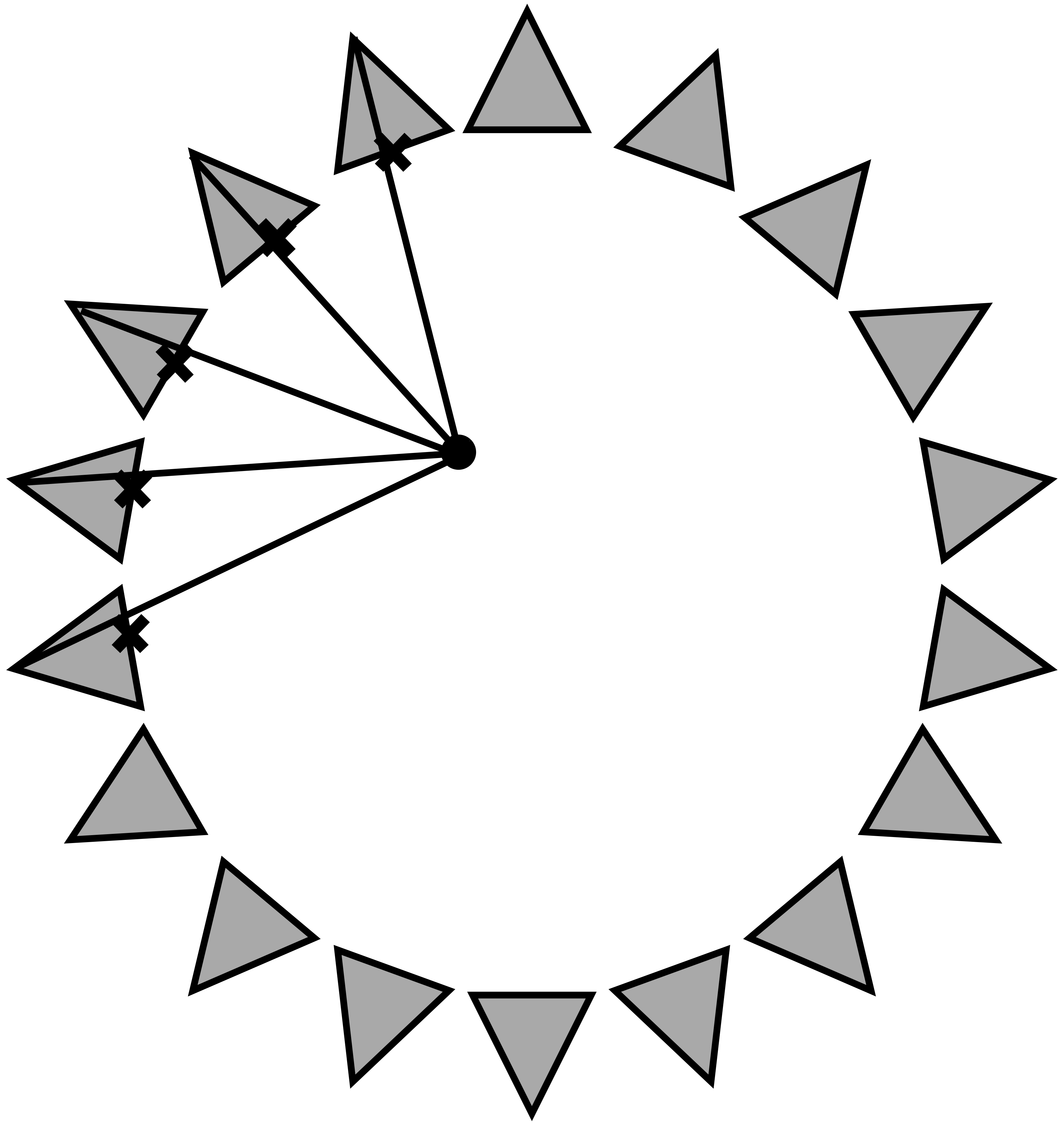


Global alignment

Global Alignment

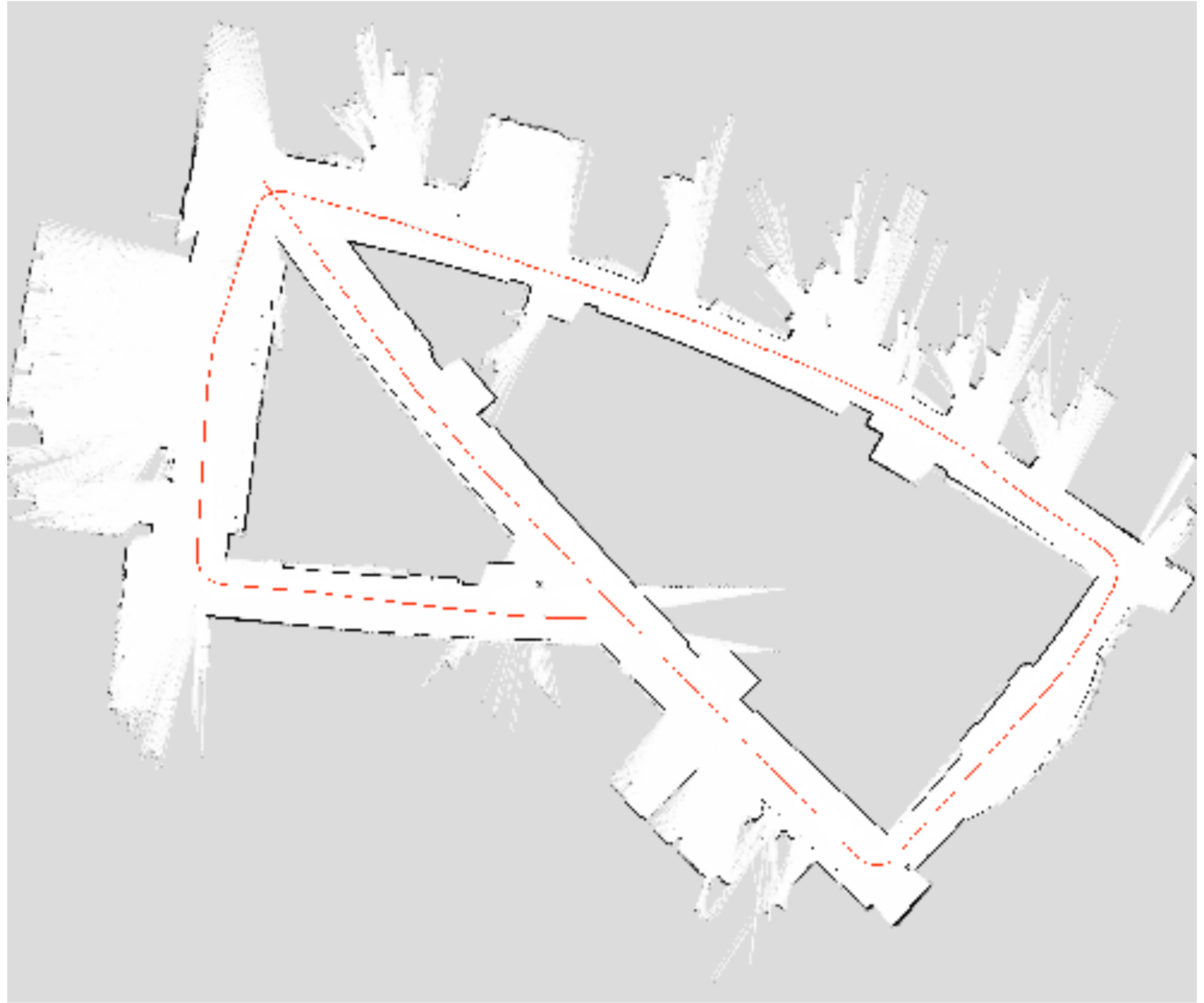
- Concatenation of pairwise R, t estimates results in drift, e.g.,


Pairwise alignment


Global alignment

Global Alignment

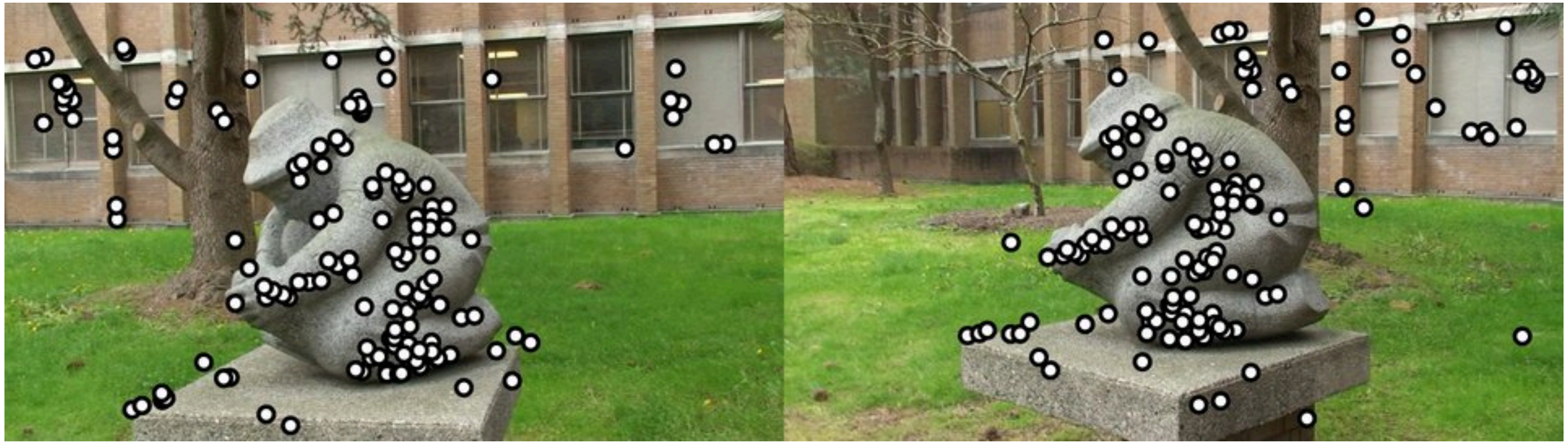
- In robotic navigation frame-frame alignment also causes drift



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

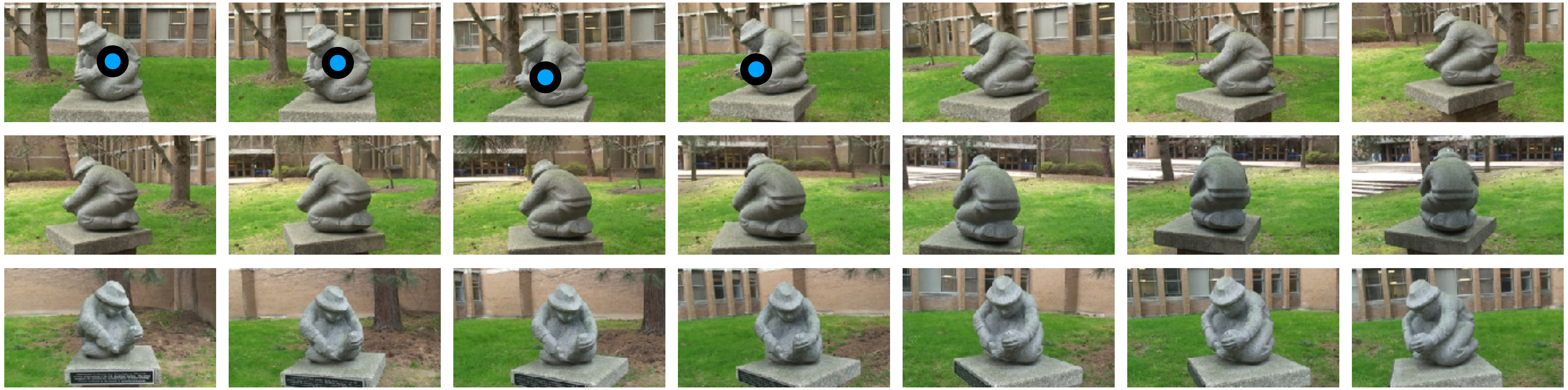
Feature Tracking

- 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

Feature Tracking

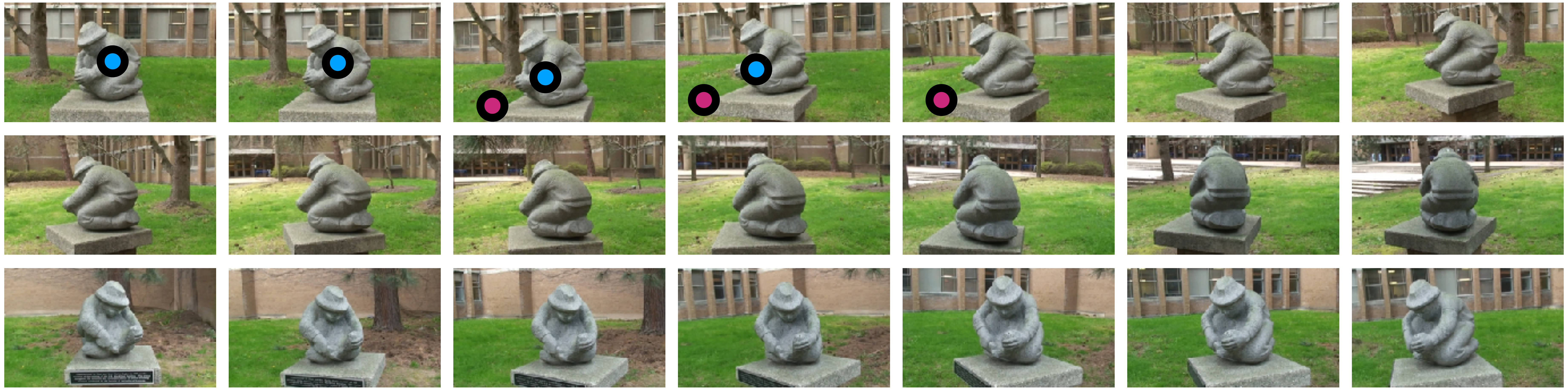
- Form feature tracks by combining pairwise feature matches



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

Feature Tracking

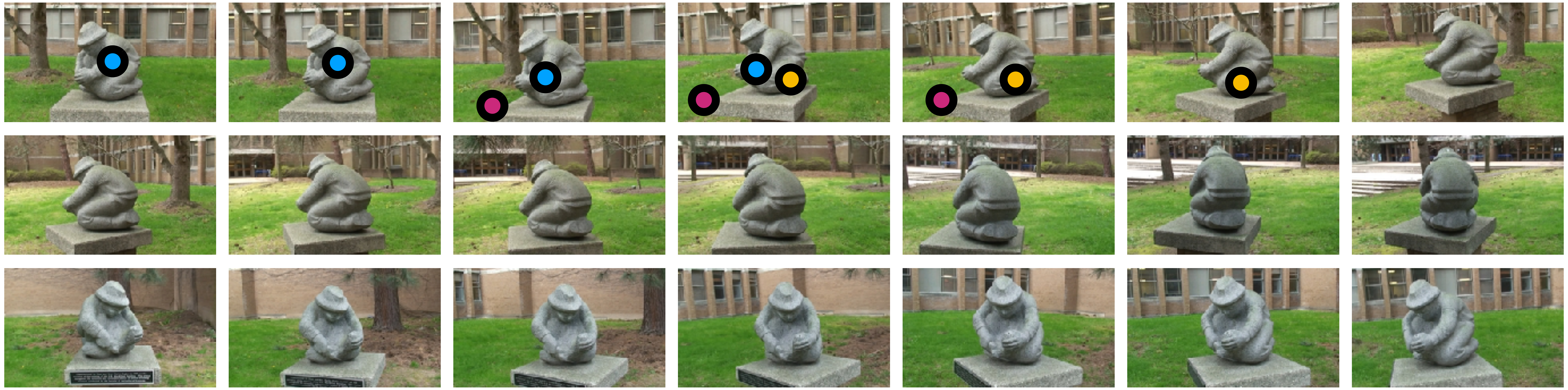
- Form feature tracks by combining pairwise feature matches



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

Feature Tracking

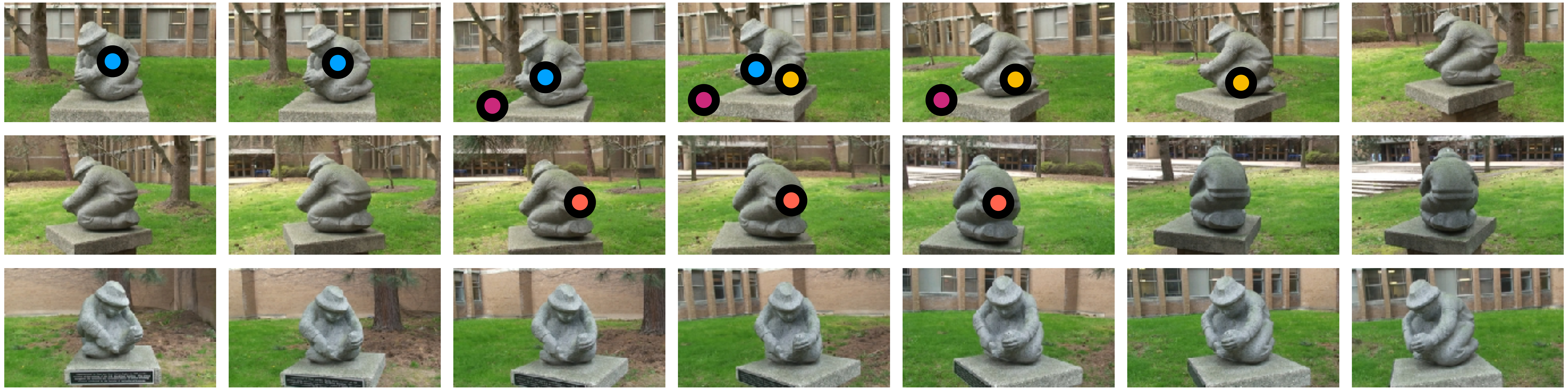
- Form feature tracks by combining pairwise feature matches



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

Feature Tracking

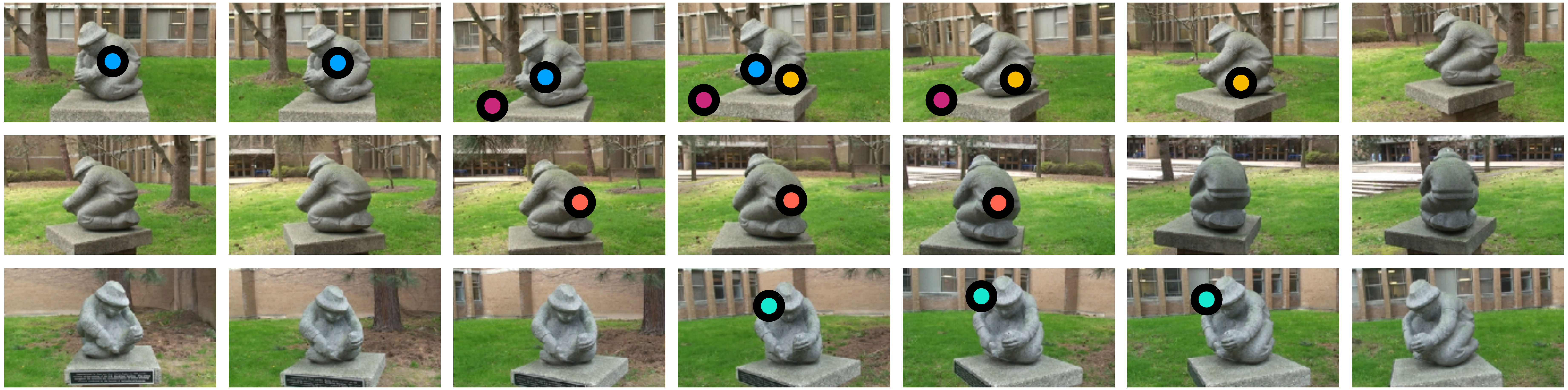
- Form feature tracks by combining pairwise feature matches



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

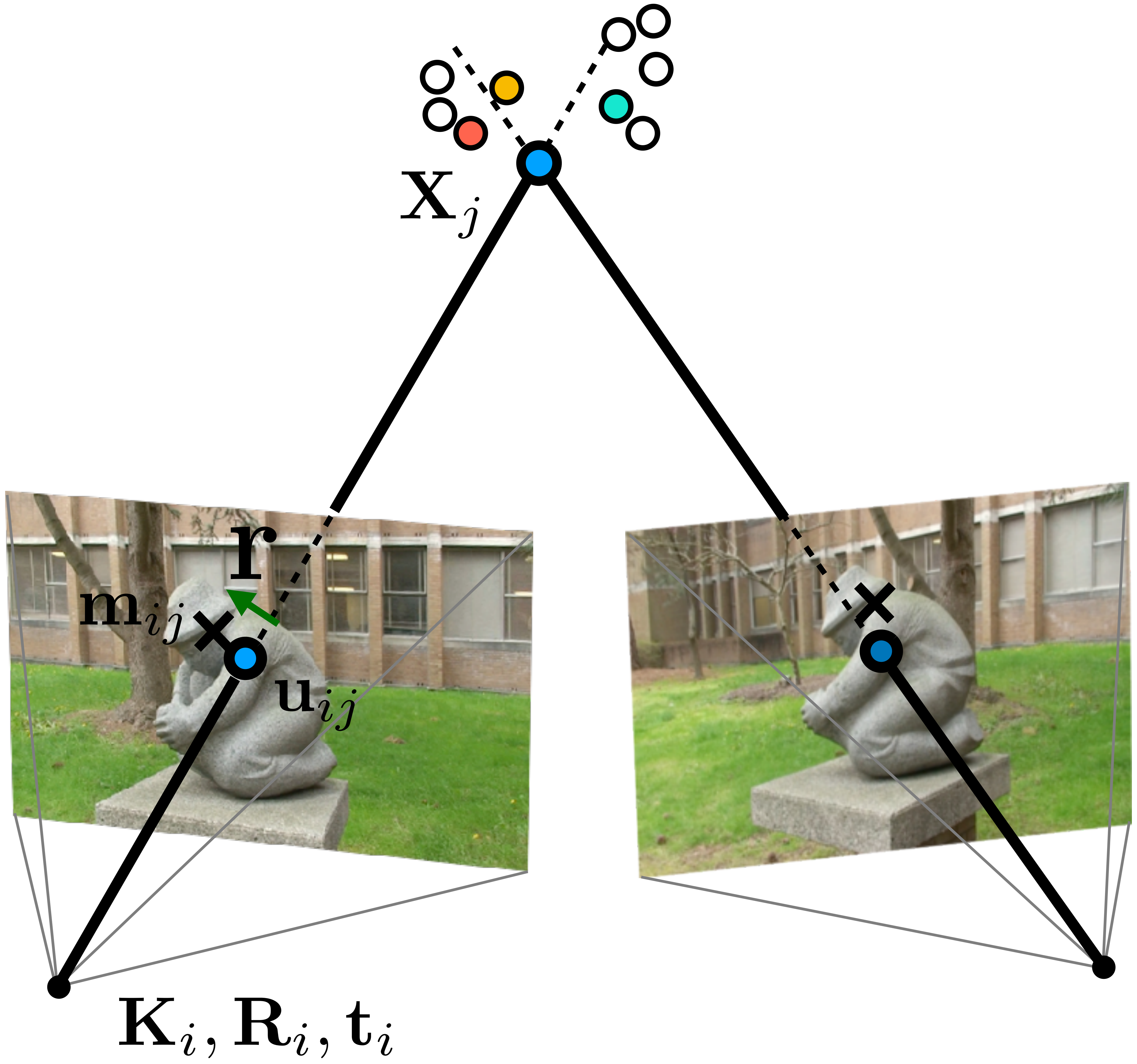
Feature Tracking

- 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)

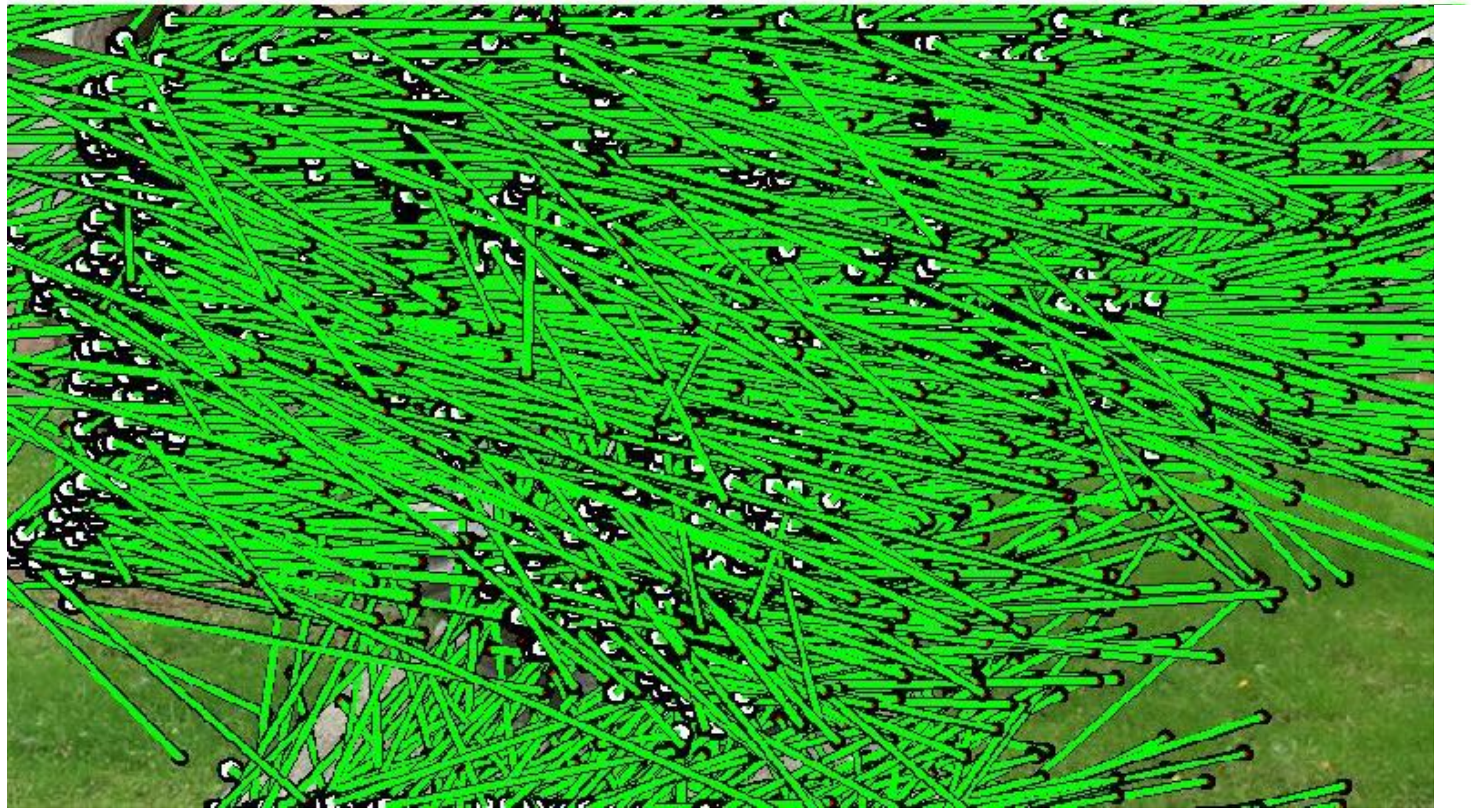
Bundle Adjustment

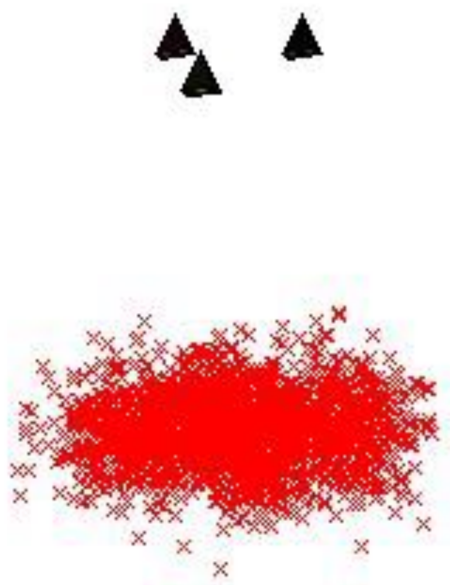
- Initialization with 3 views

Joint optimization of cameras and structure

Bundle Adjustment

- Initialization with 3 views

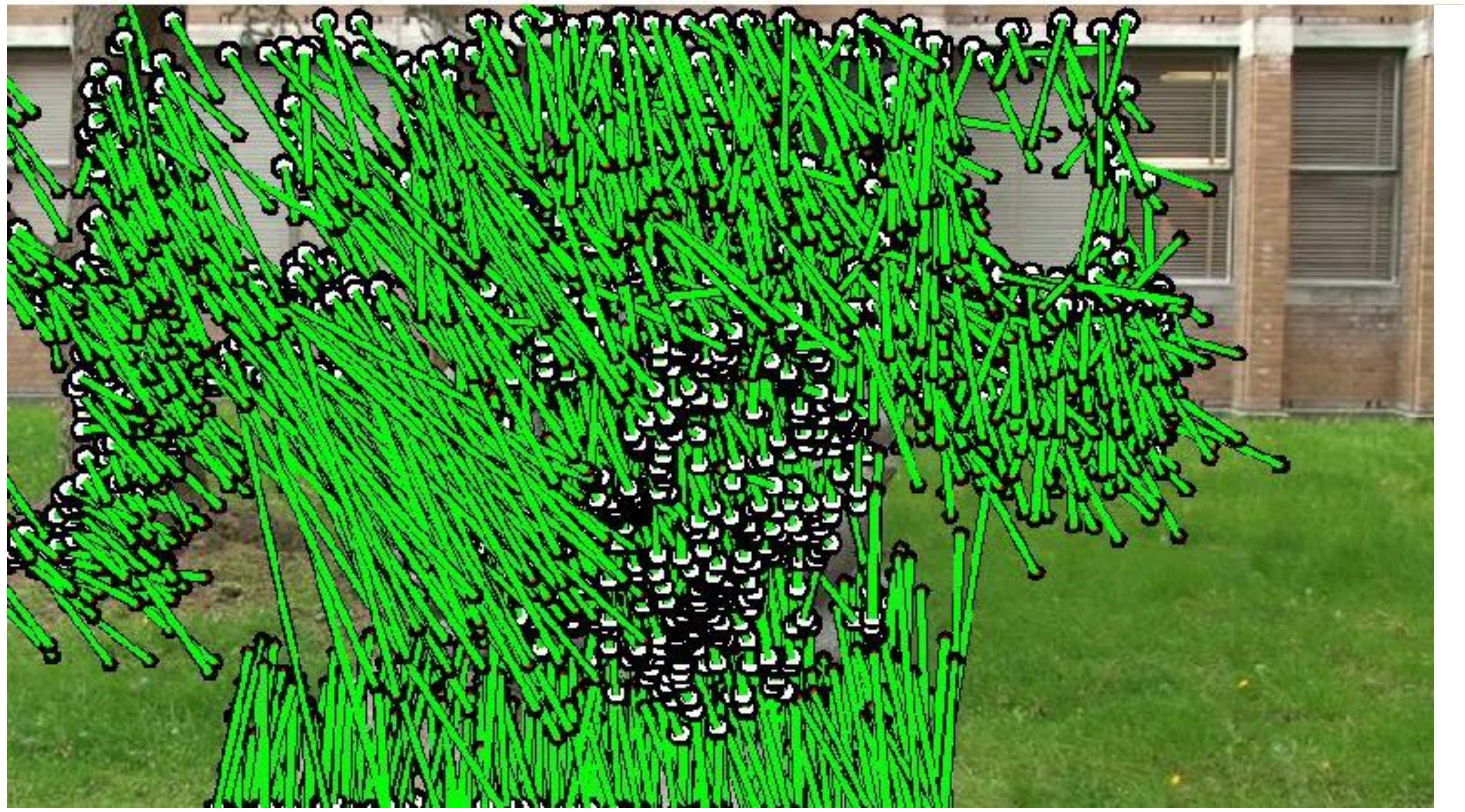


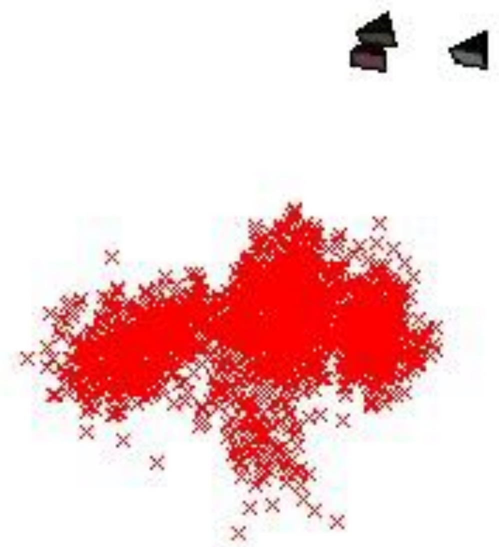


Joint optimization of cameras and structure

Bundle Adjustment

- Initialization with 3 views

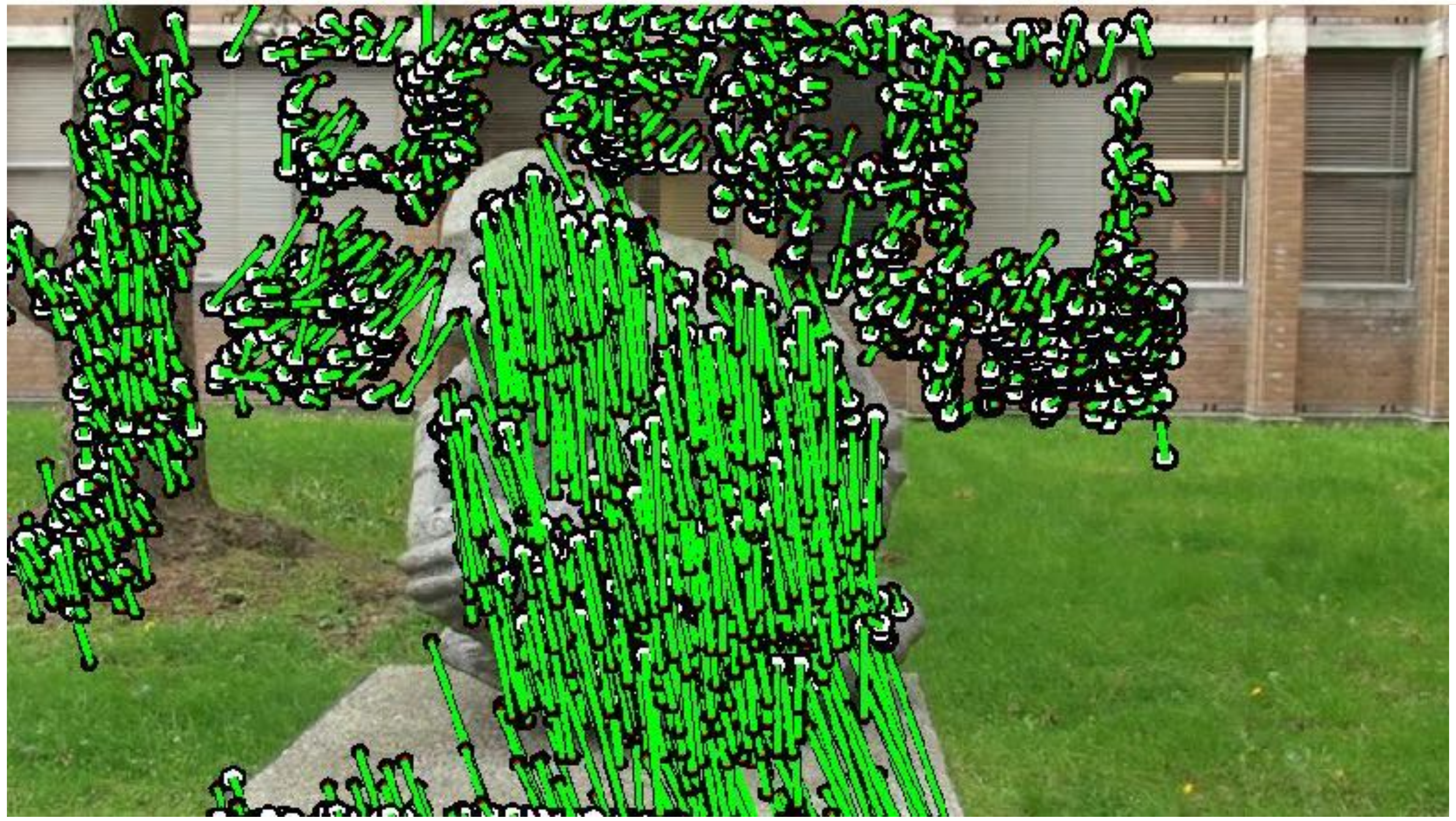




Joint optimization of cameras and structure

Bundle Adjustment

- Initialization with 3 views



Joint optimization of cameras and structure

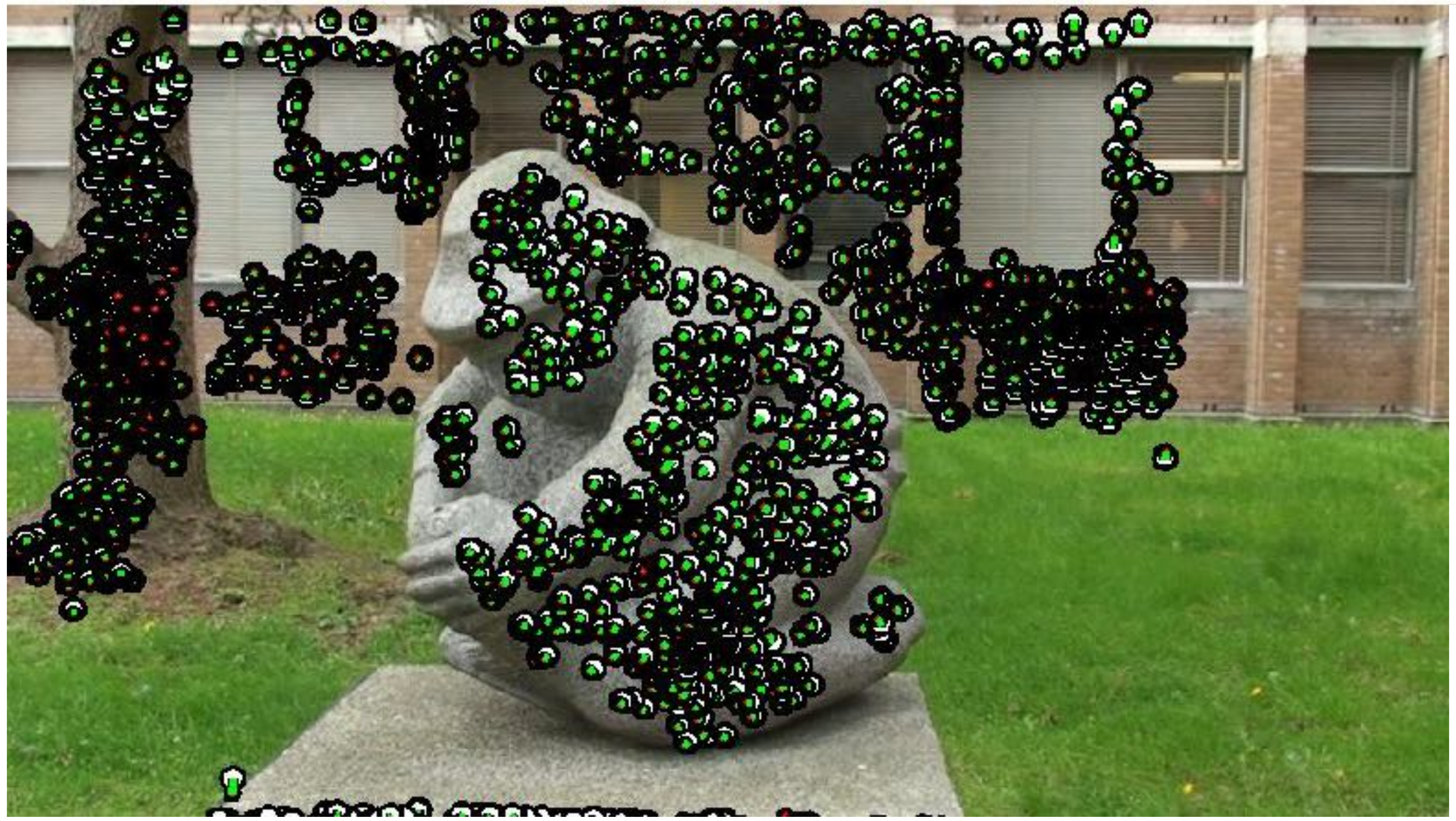
Bundle Adjustment

- Initialization with 3 views

Joint optimization of cameras and structure

Bundle Adjustment

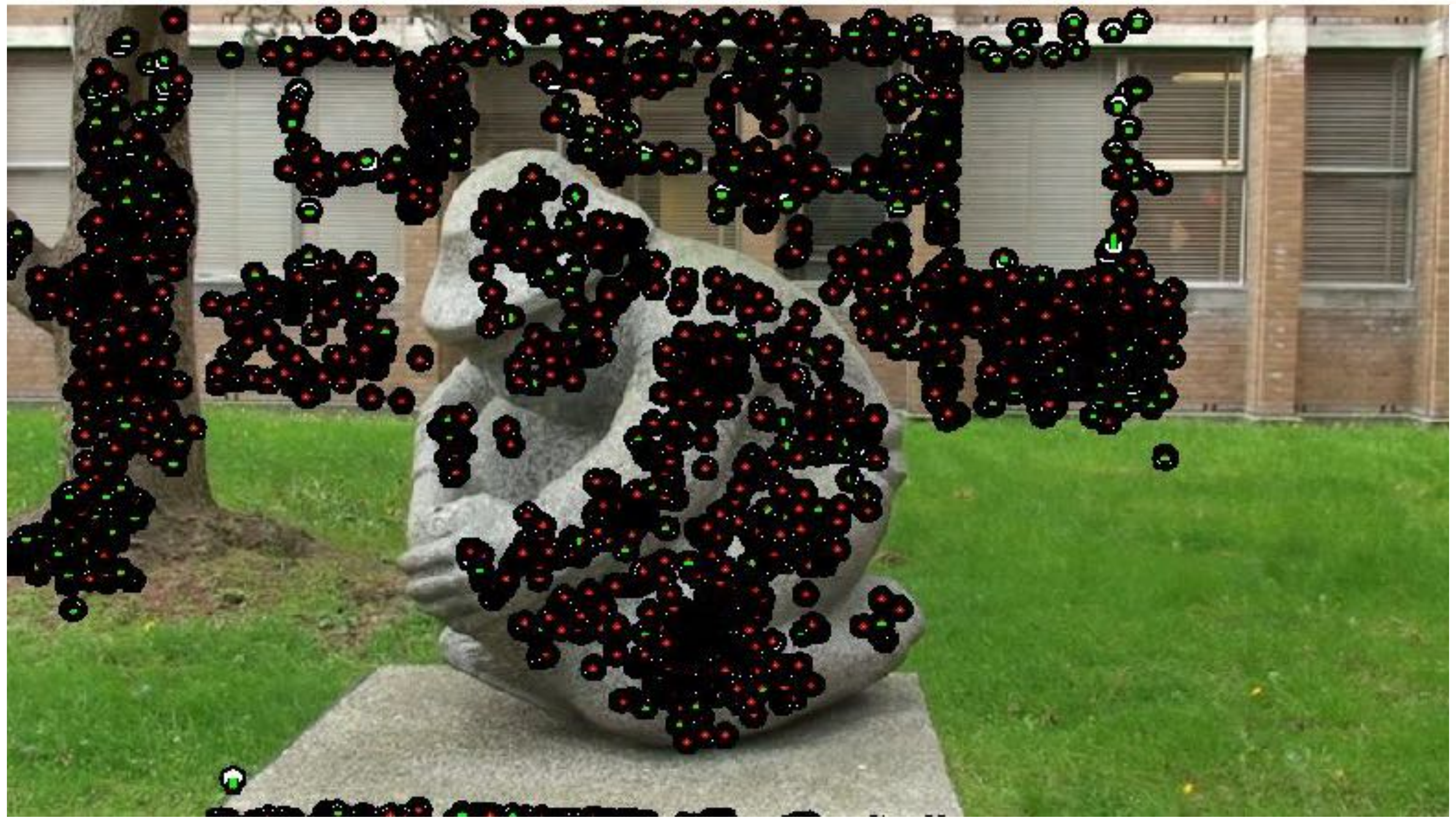
- Initialization with 3 views



Joint optimization of cameras and structure

Bundle Adjustment

- Initialization with 3 views



Joint optimization of cameras and structure

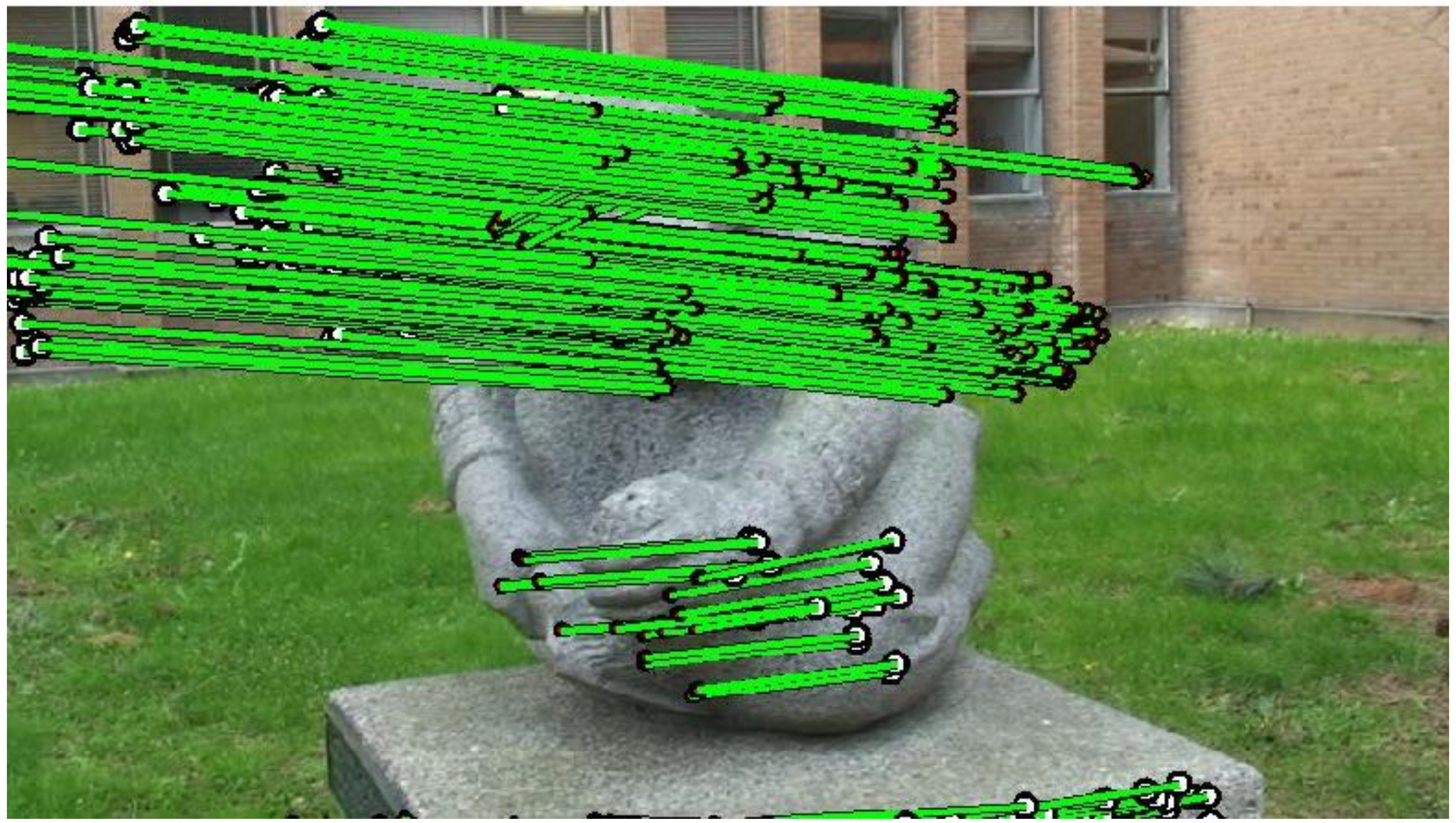
Bundle Adjustment

- Add camera 4

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

Bundle Adjustment

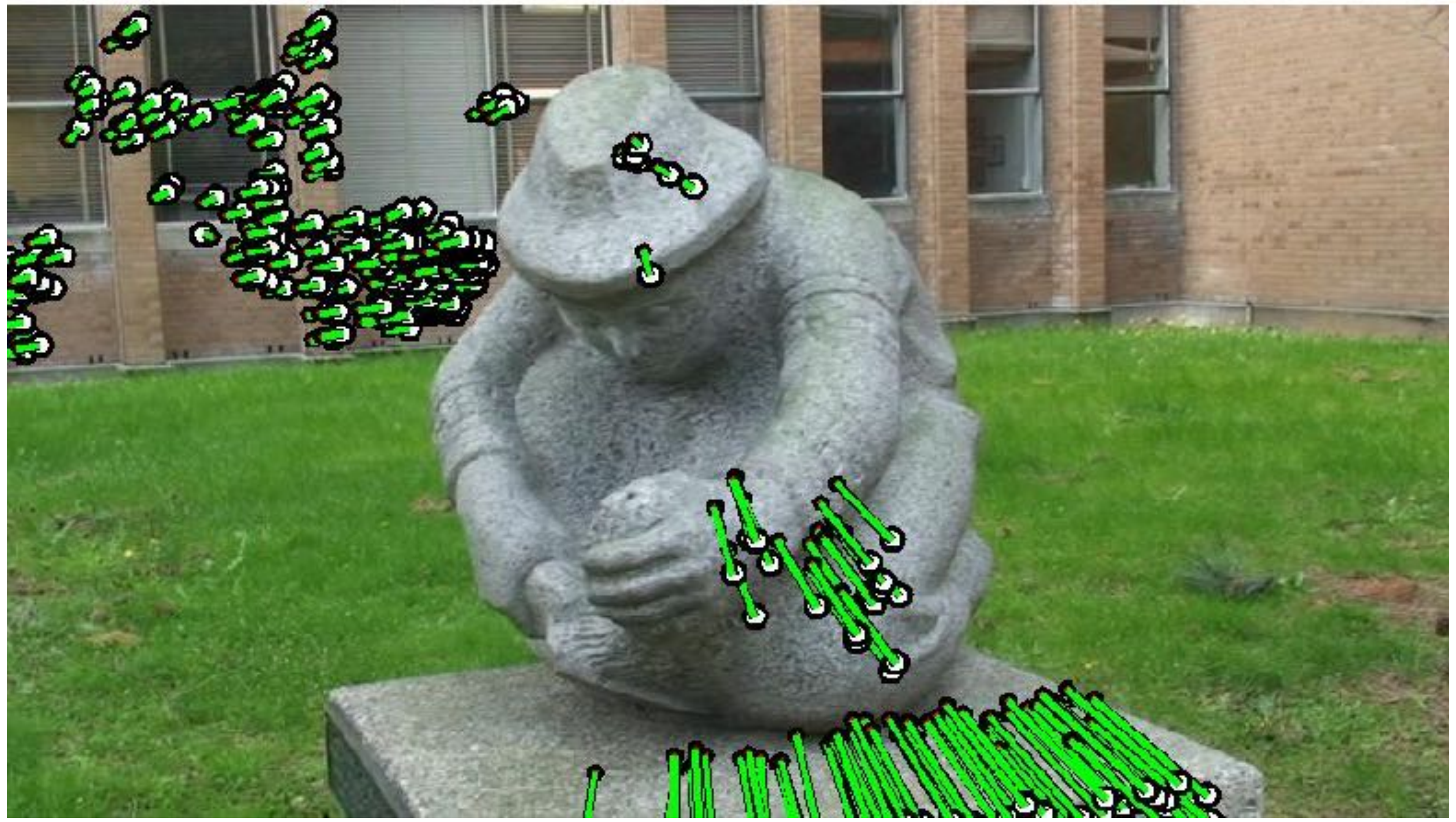
- Add camera 4



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

Bundle Adjustment

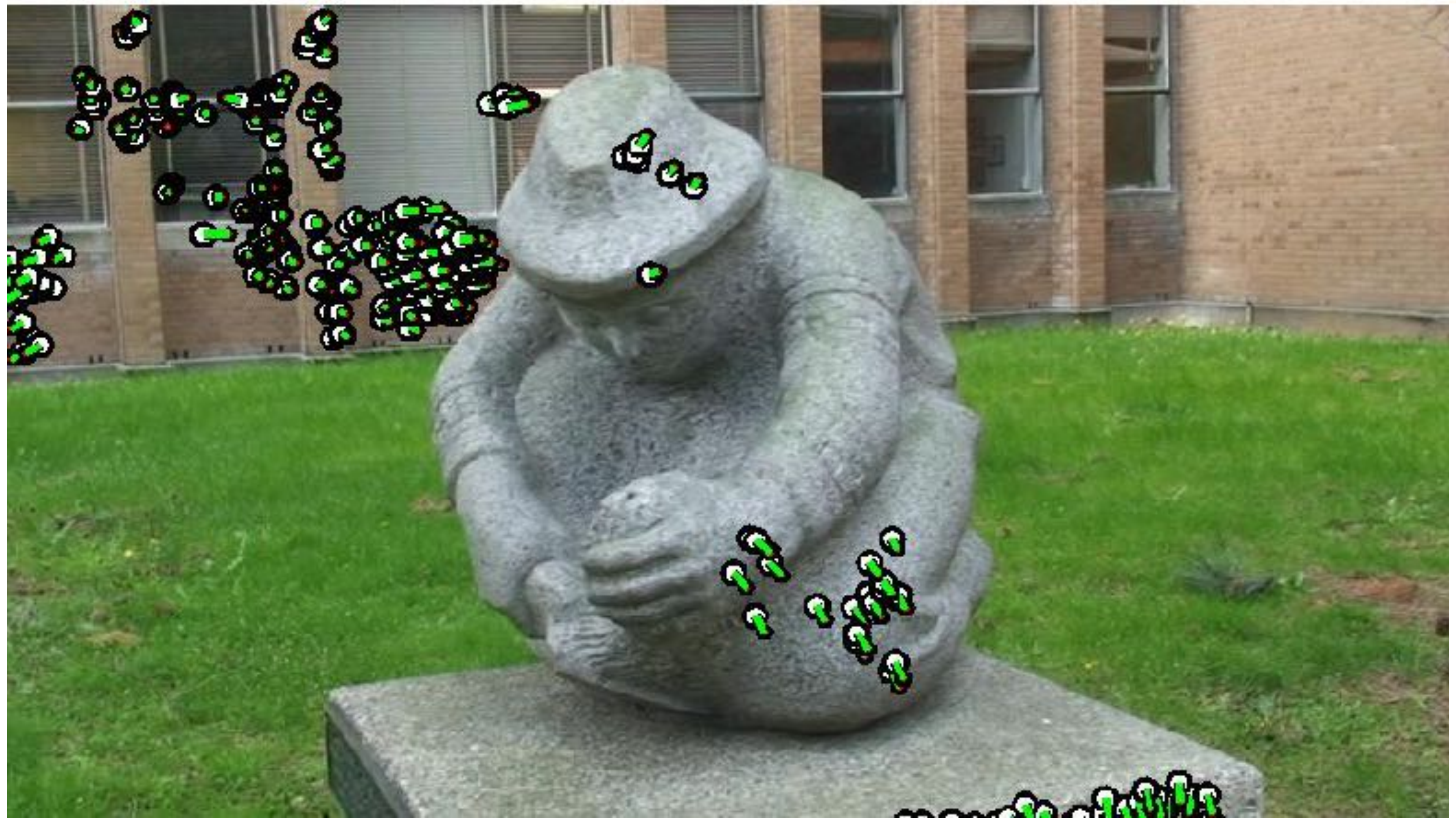
- Add camera 4



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

Bundle Adjustment

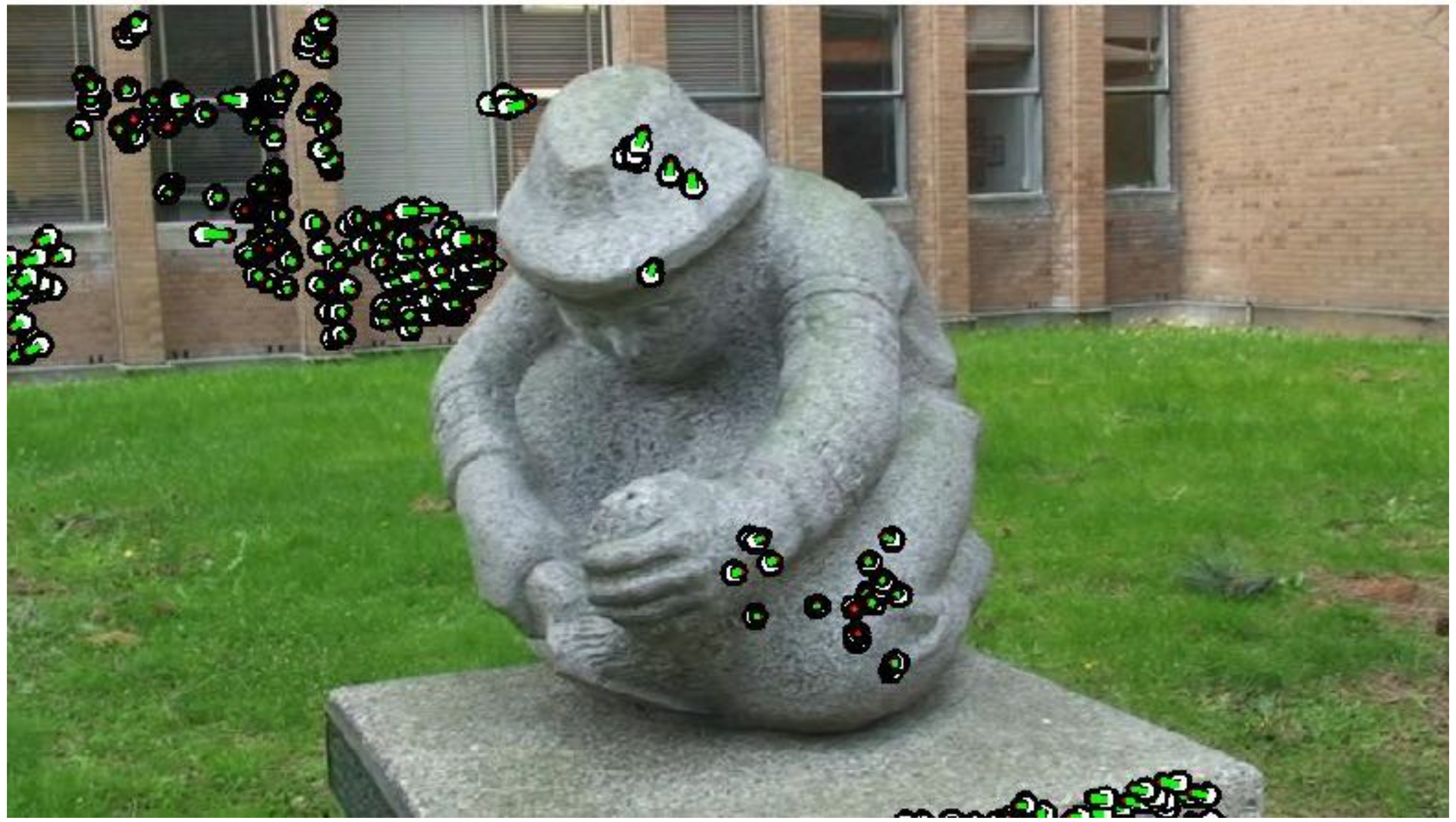
- Add camera 4



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

Bundle Adjustment

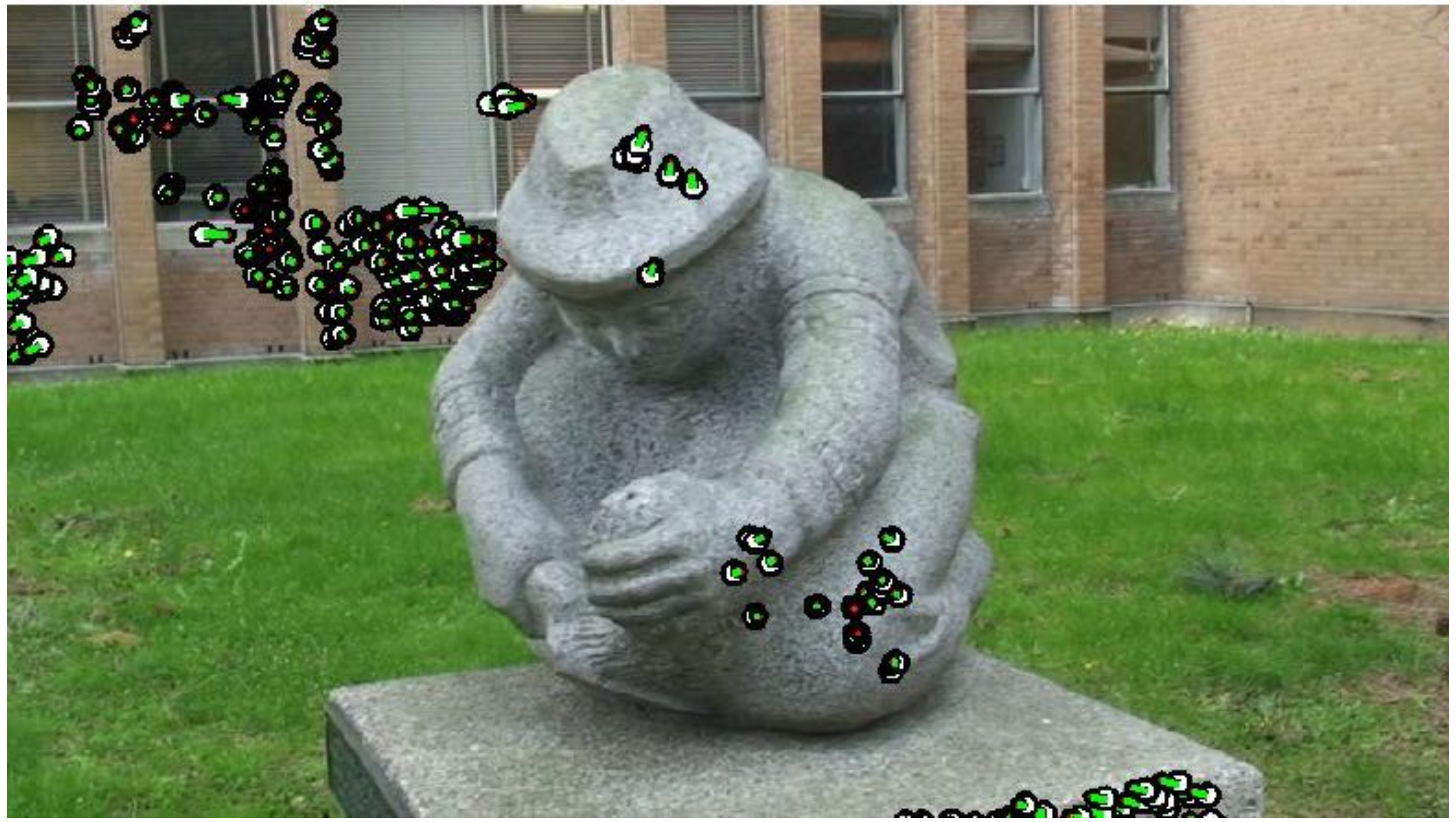
- Add camera 4



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

Bundle Adjustment

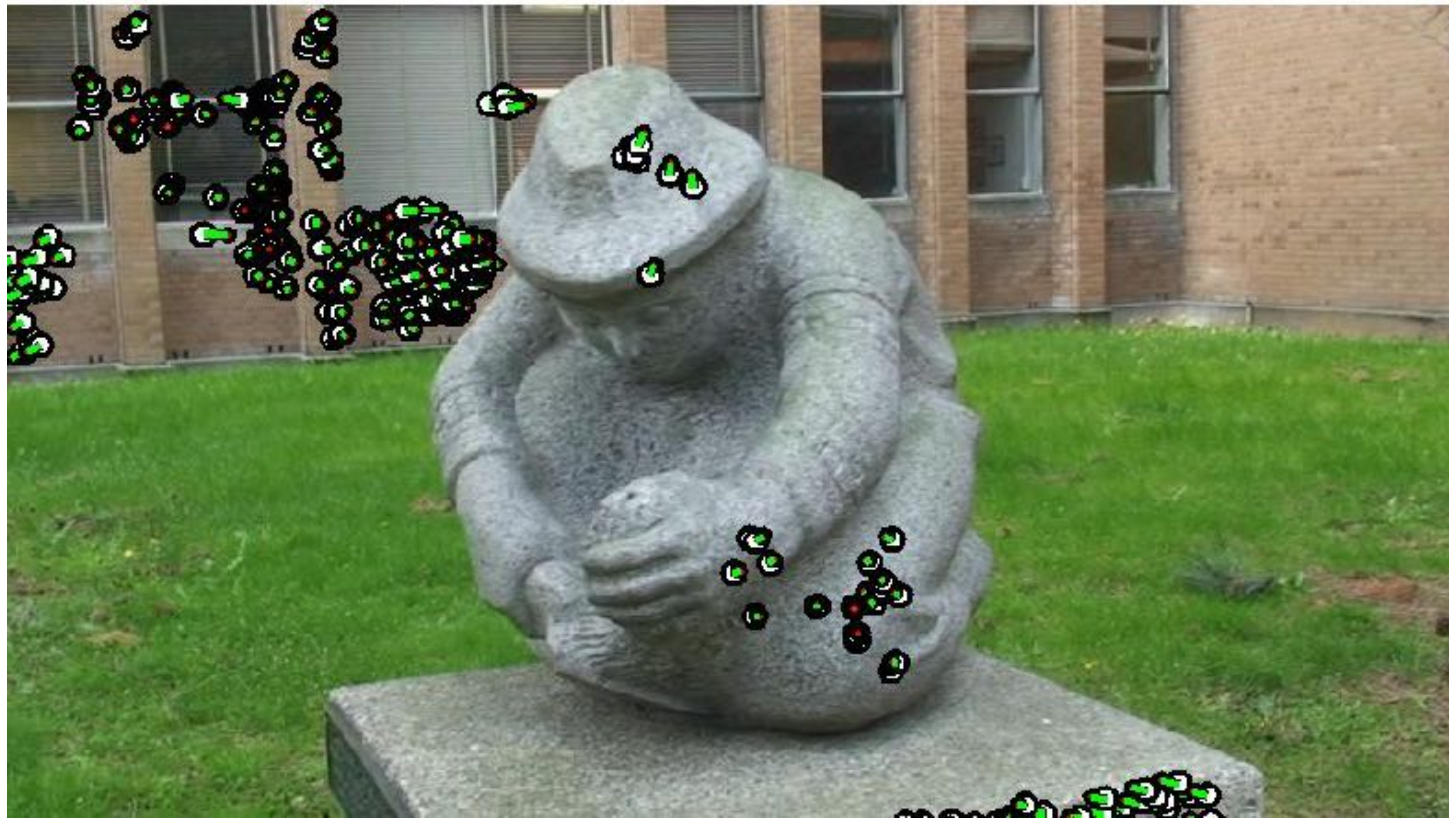
- Add camera 4



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

Bundle Adjustment

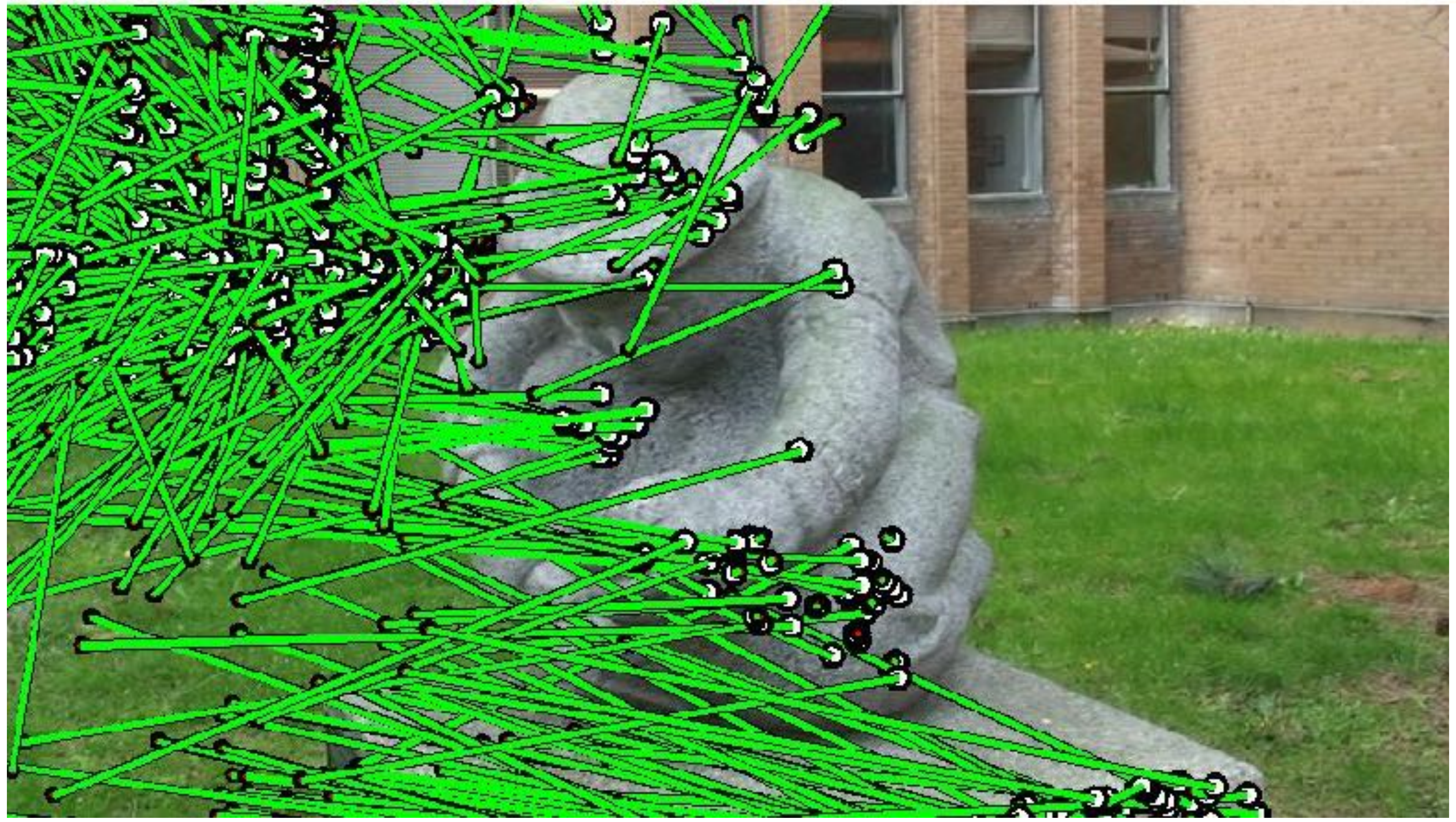
- Add camera 4



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

Bundle Adjustment

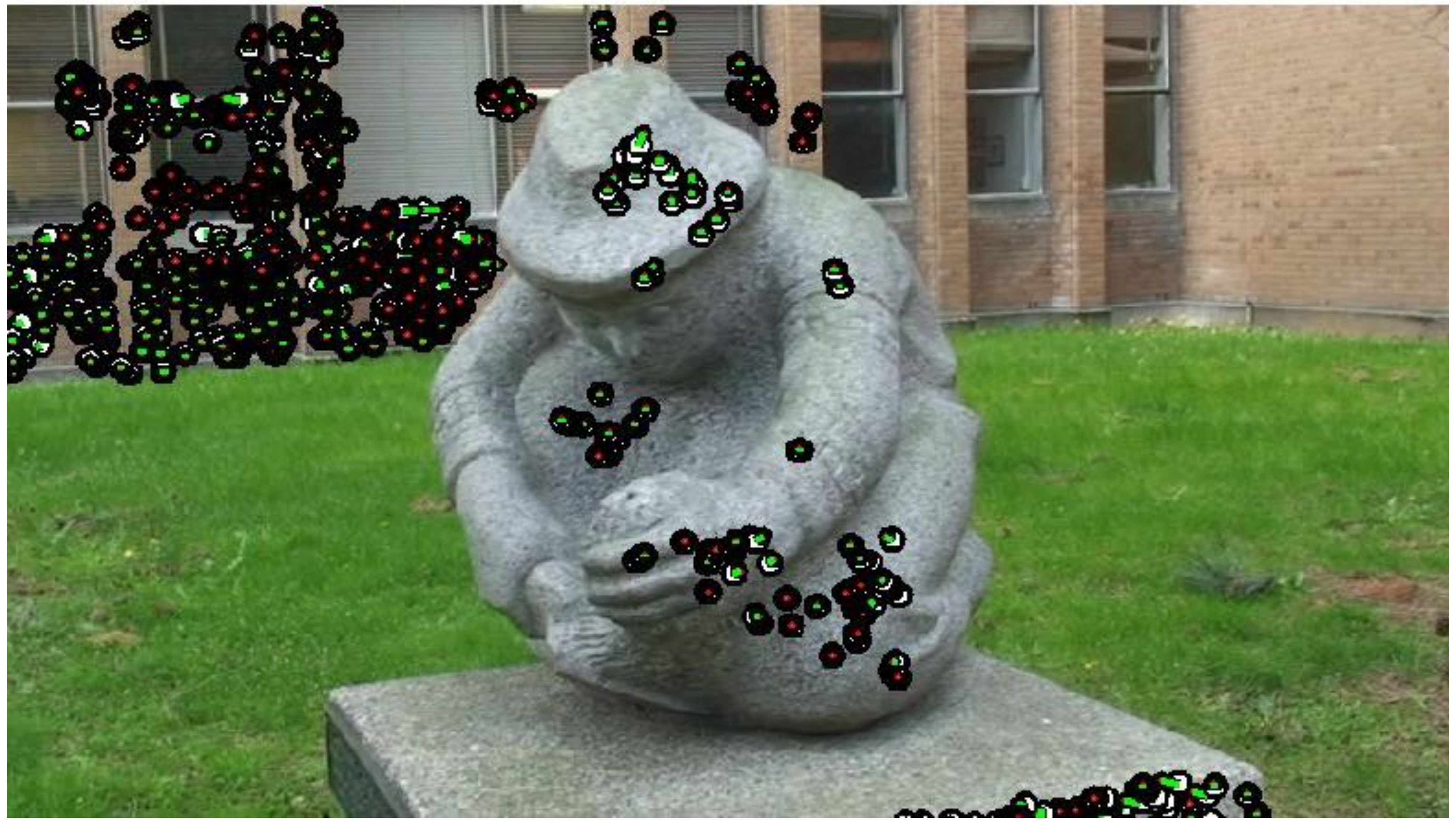
- Add camera 4

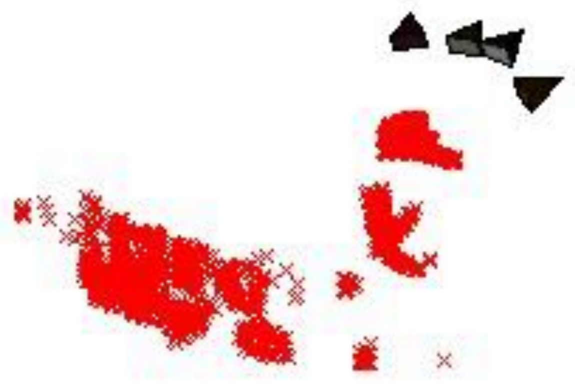


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 4

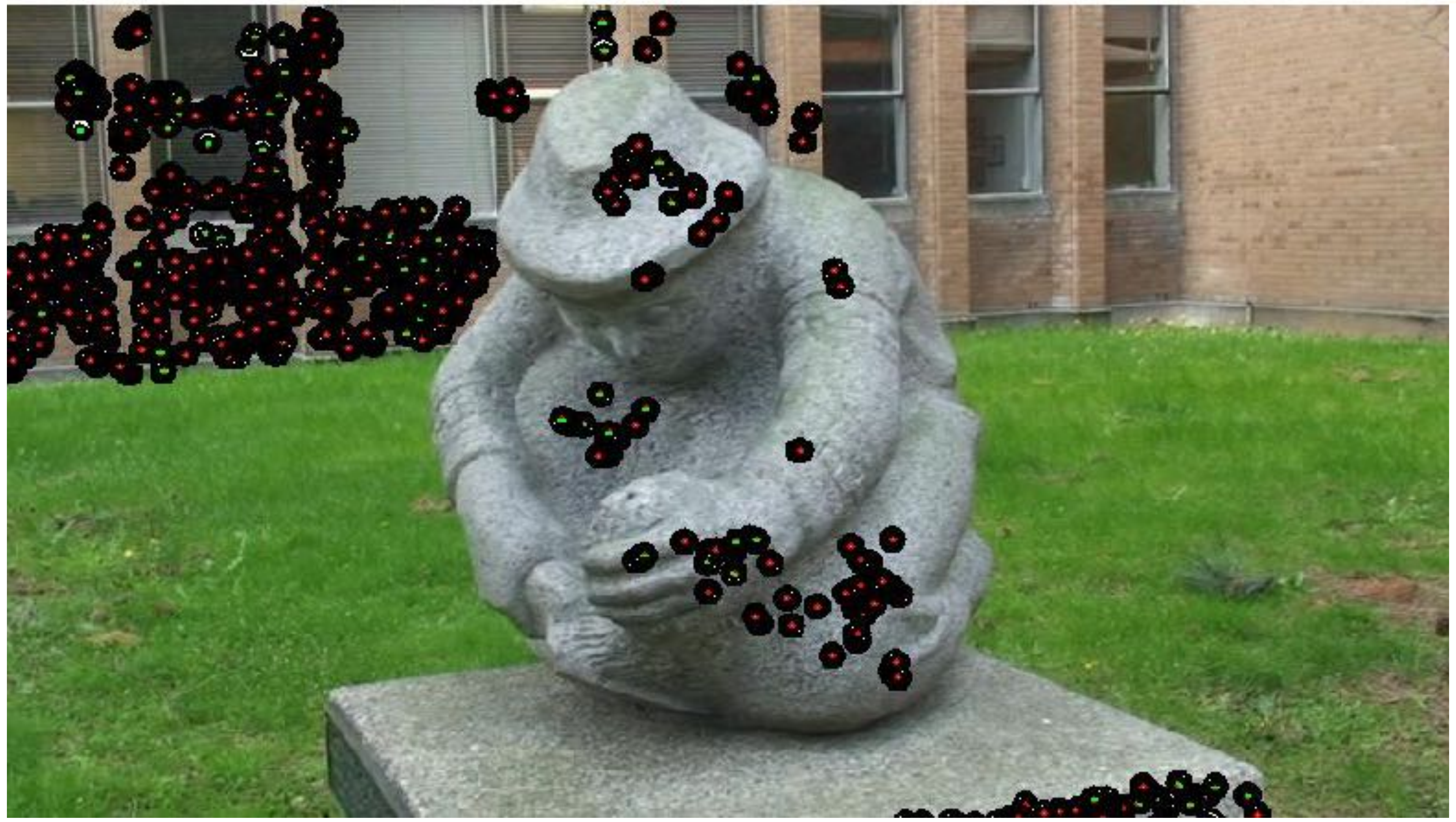




Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 4



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

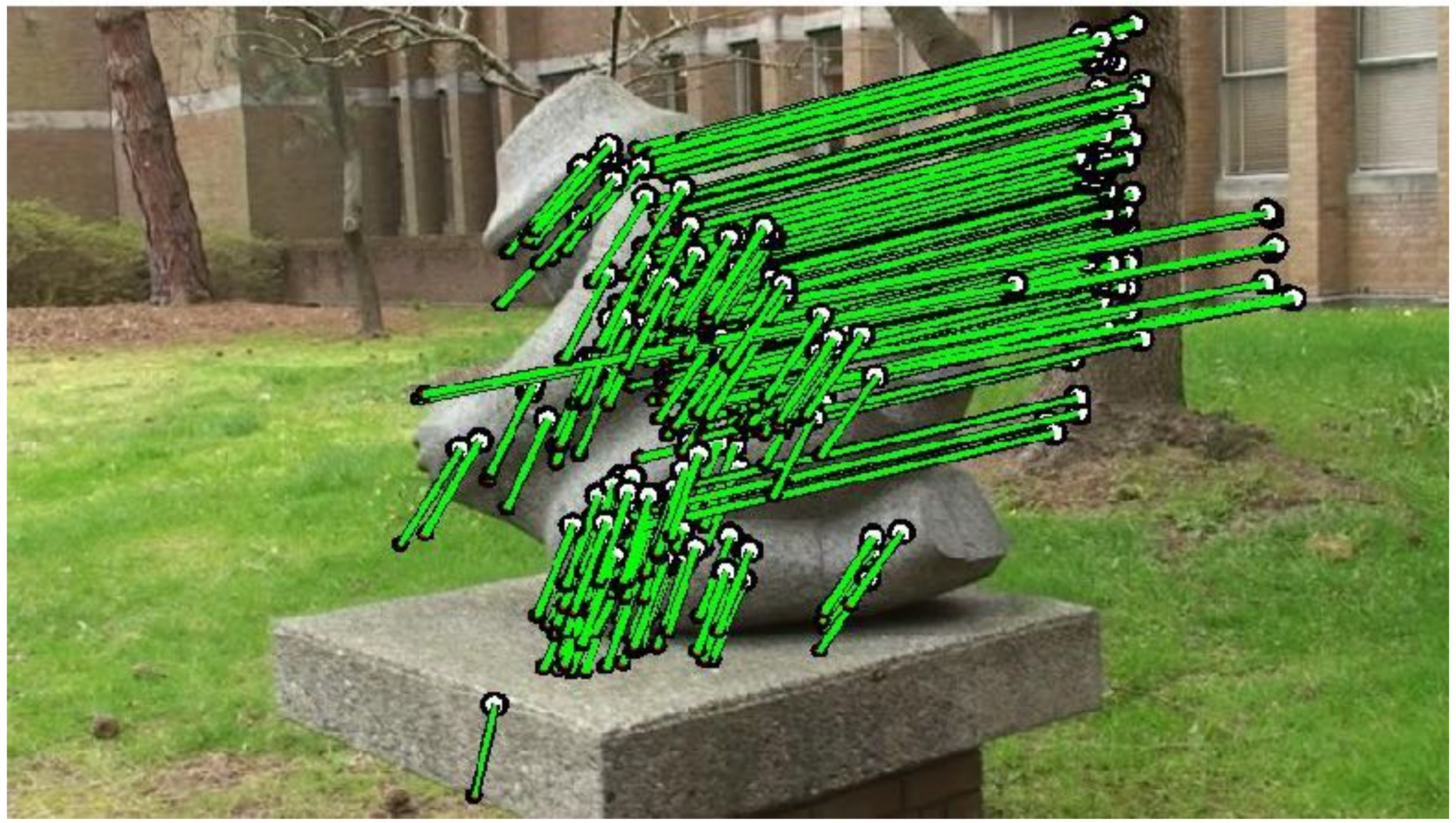
Bundle Adjustment

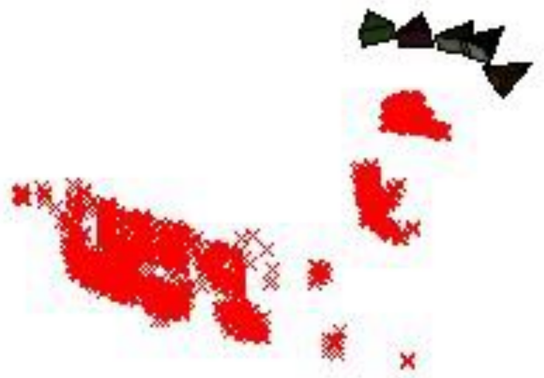
- Add camera 5

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

Bundle Adjustment

- Add camera 5

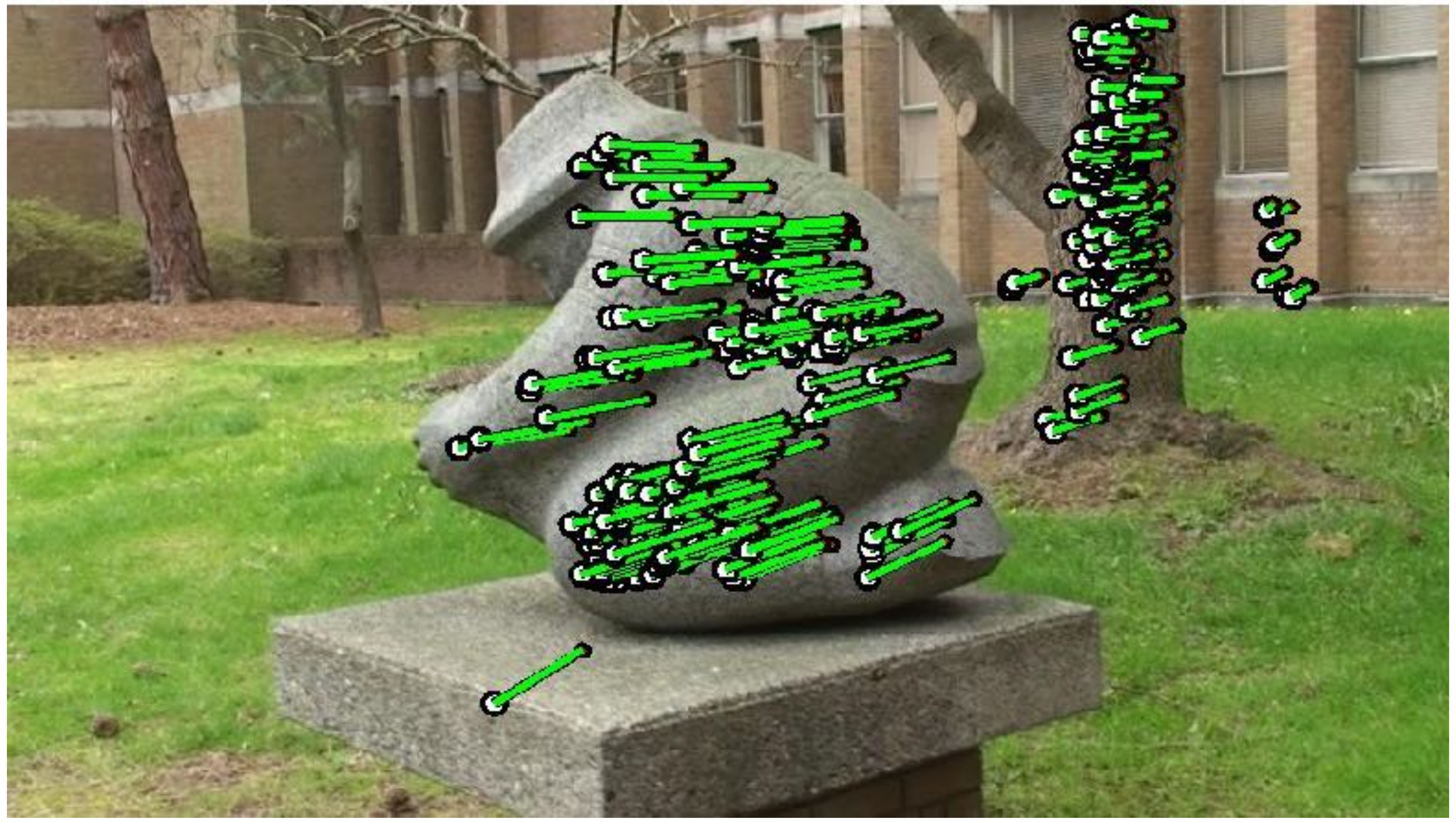




Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

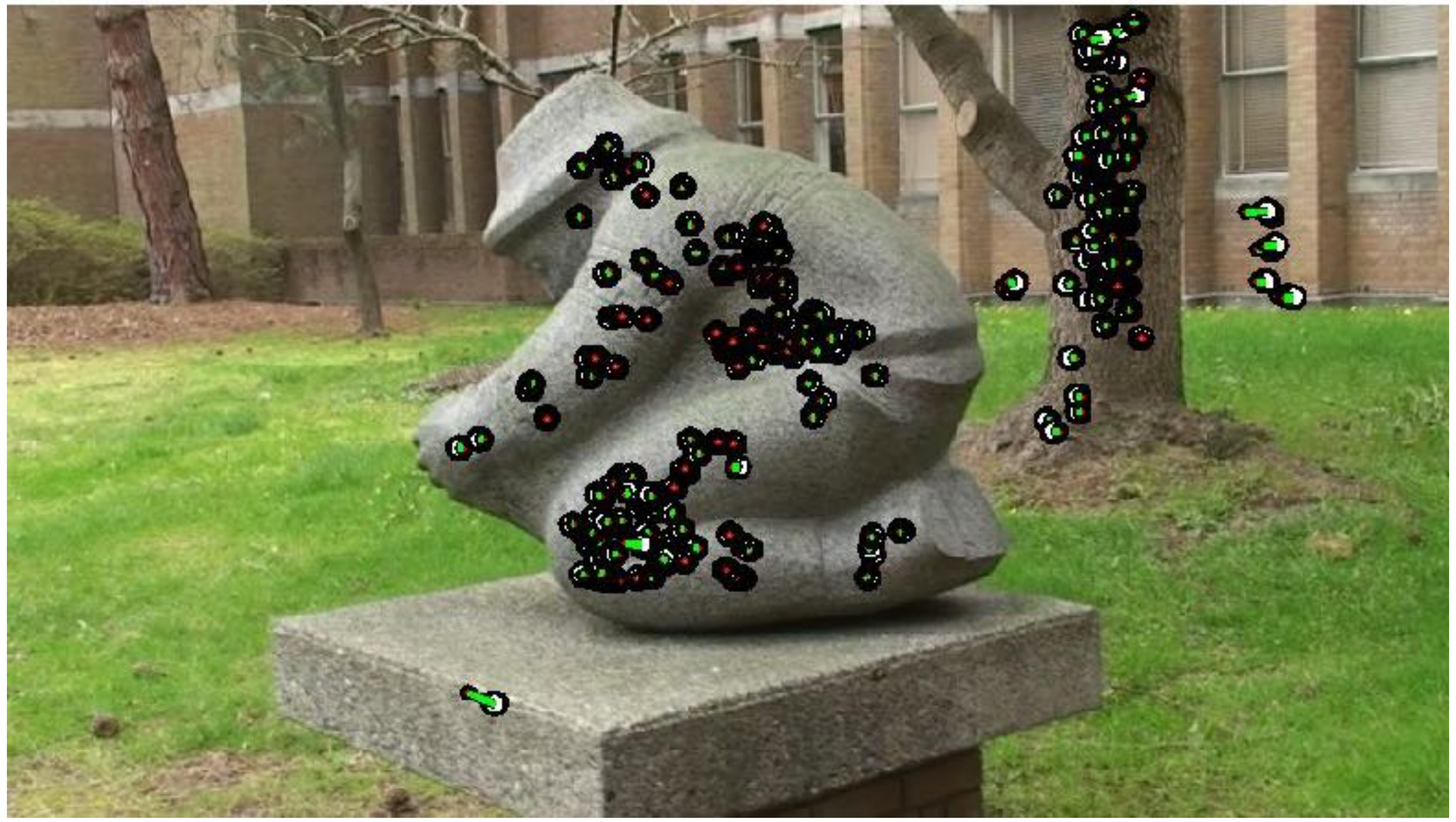
- Add camera 5

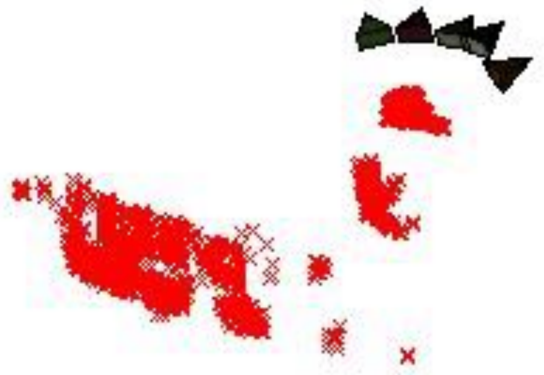


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 5

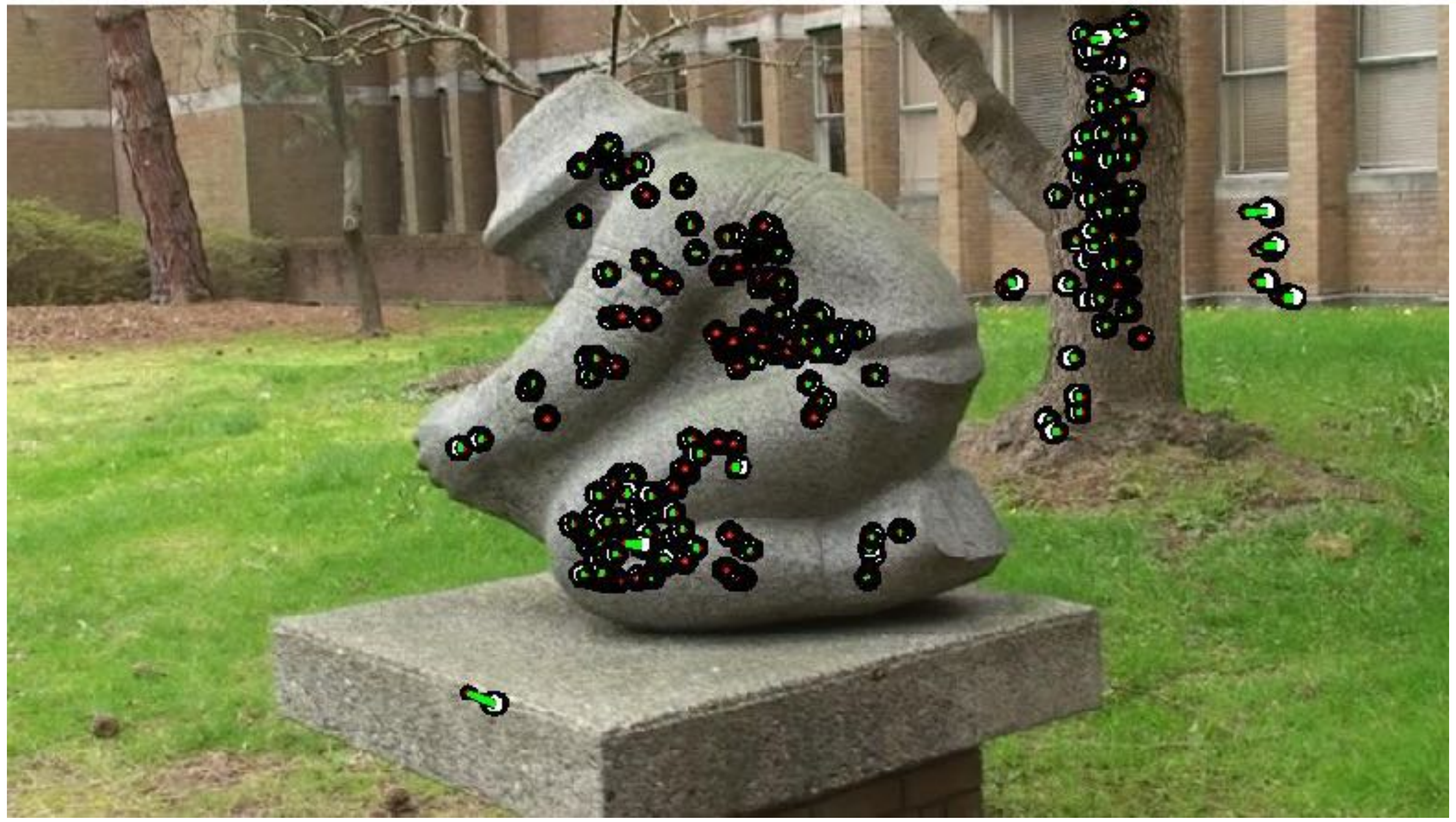




Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

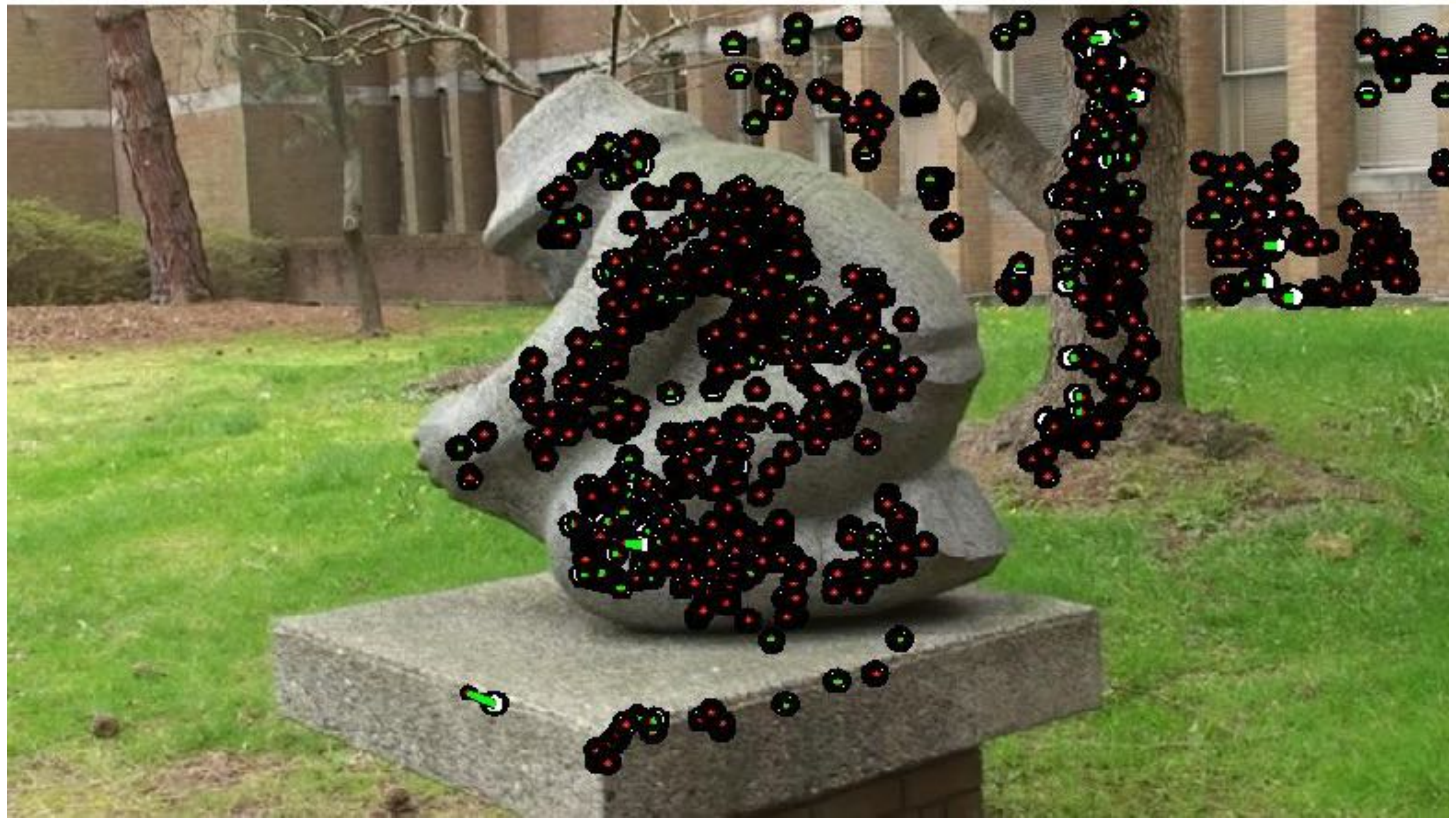
- Add camera 5

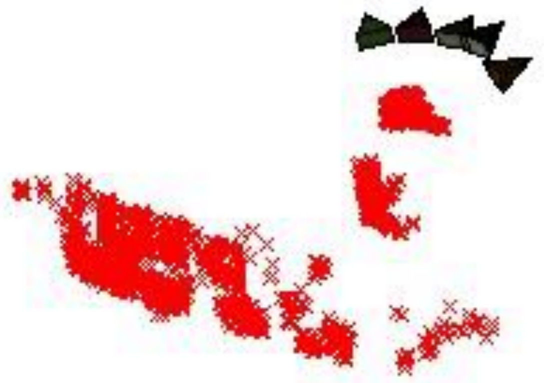


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 5

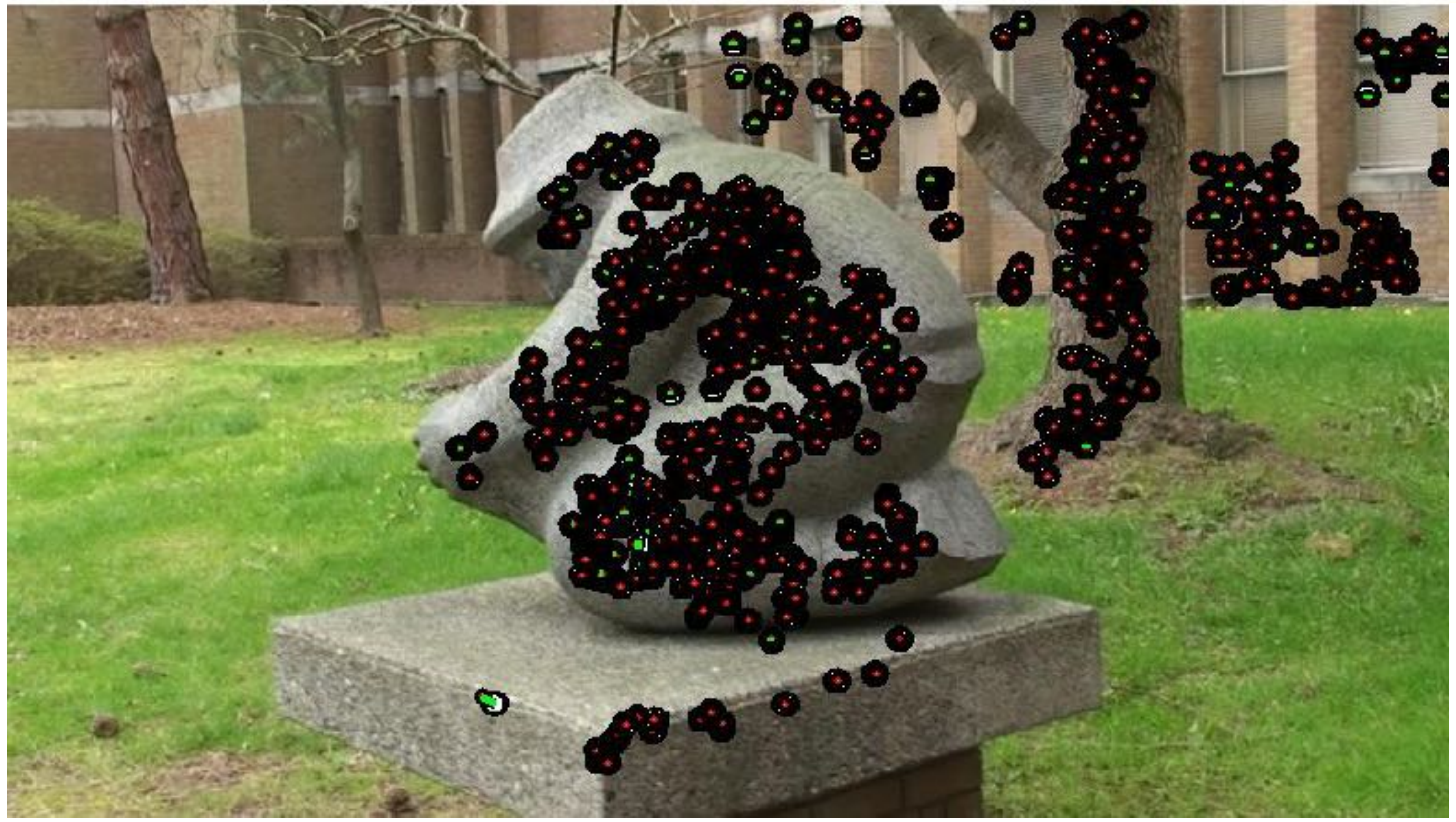


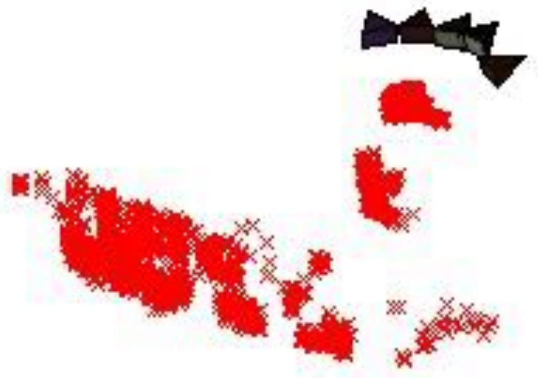


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 5





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

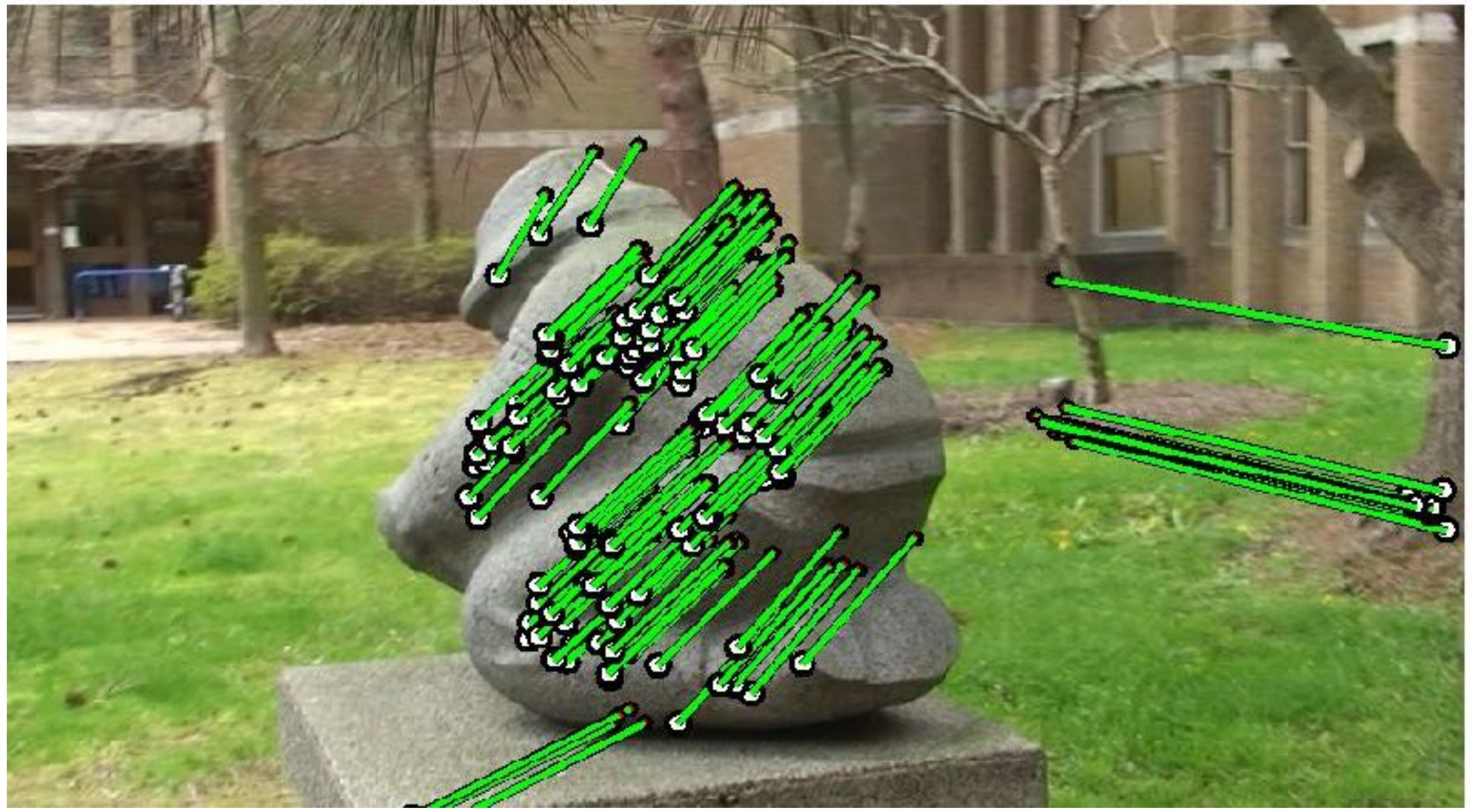
Bundle Adjustment

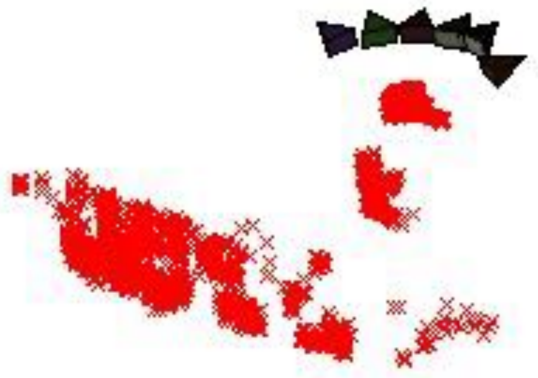
- Add camera 6

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

Bundle Adjustment

- Add camera 6

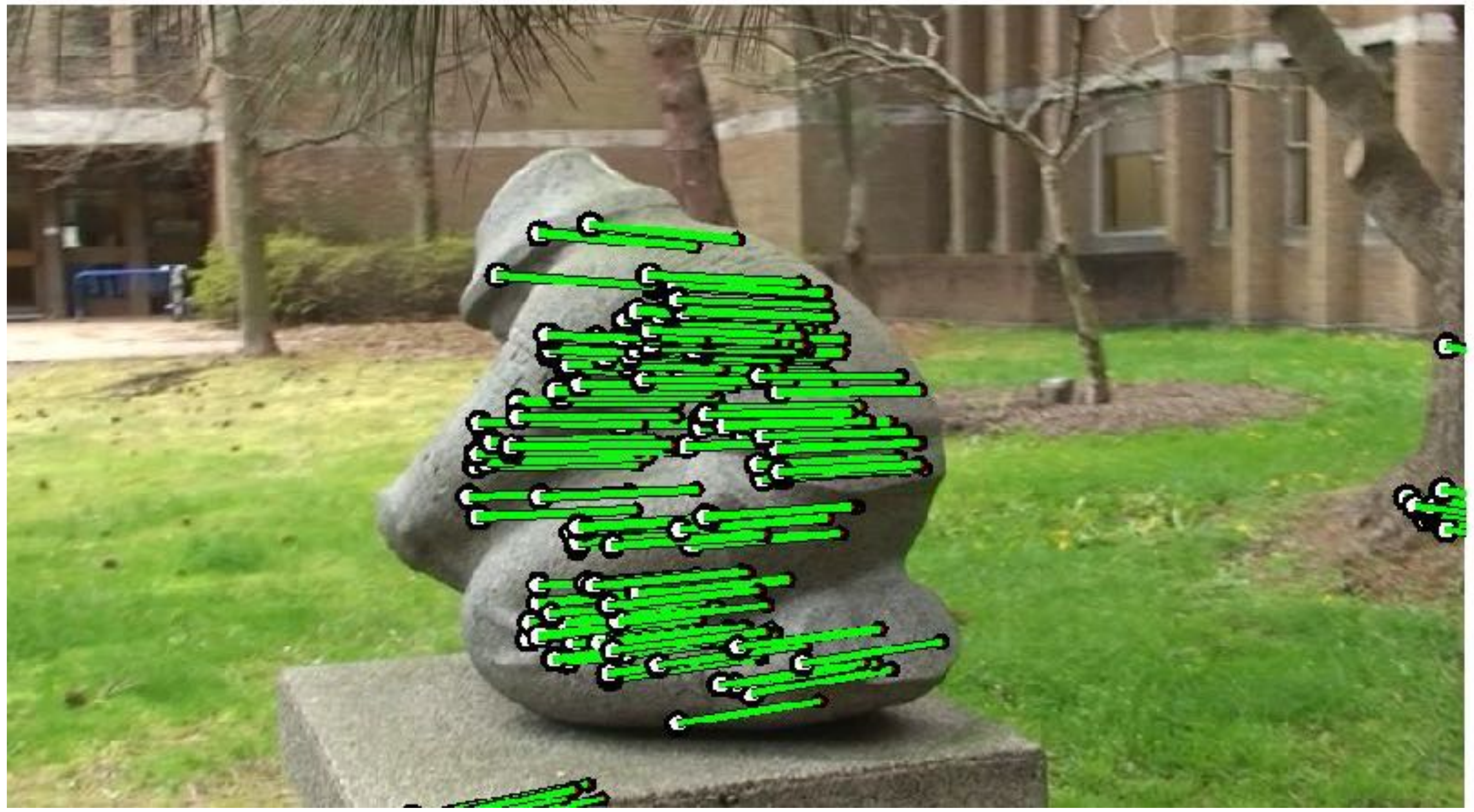


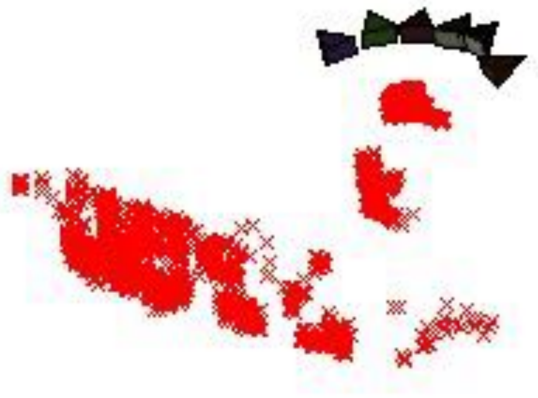


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6

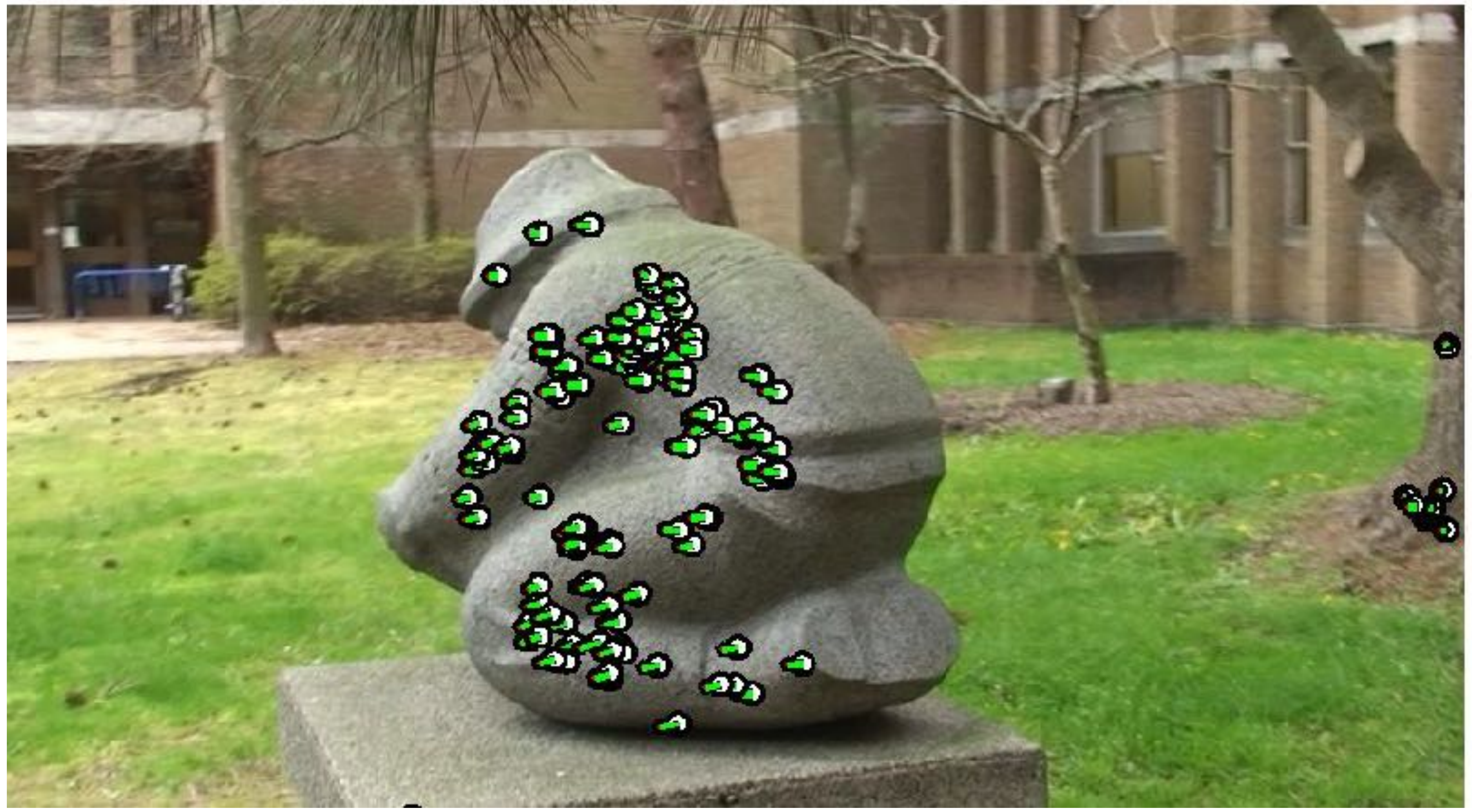


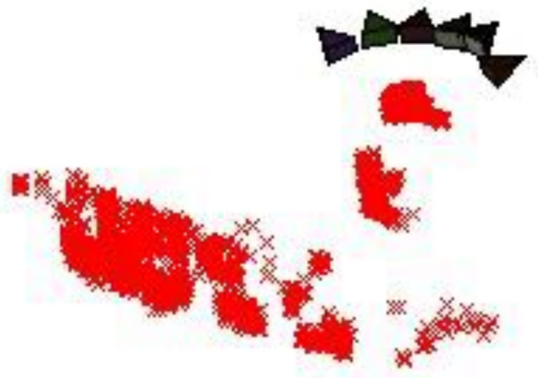


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6

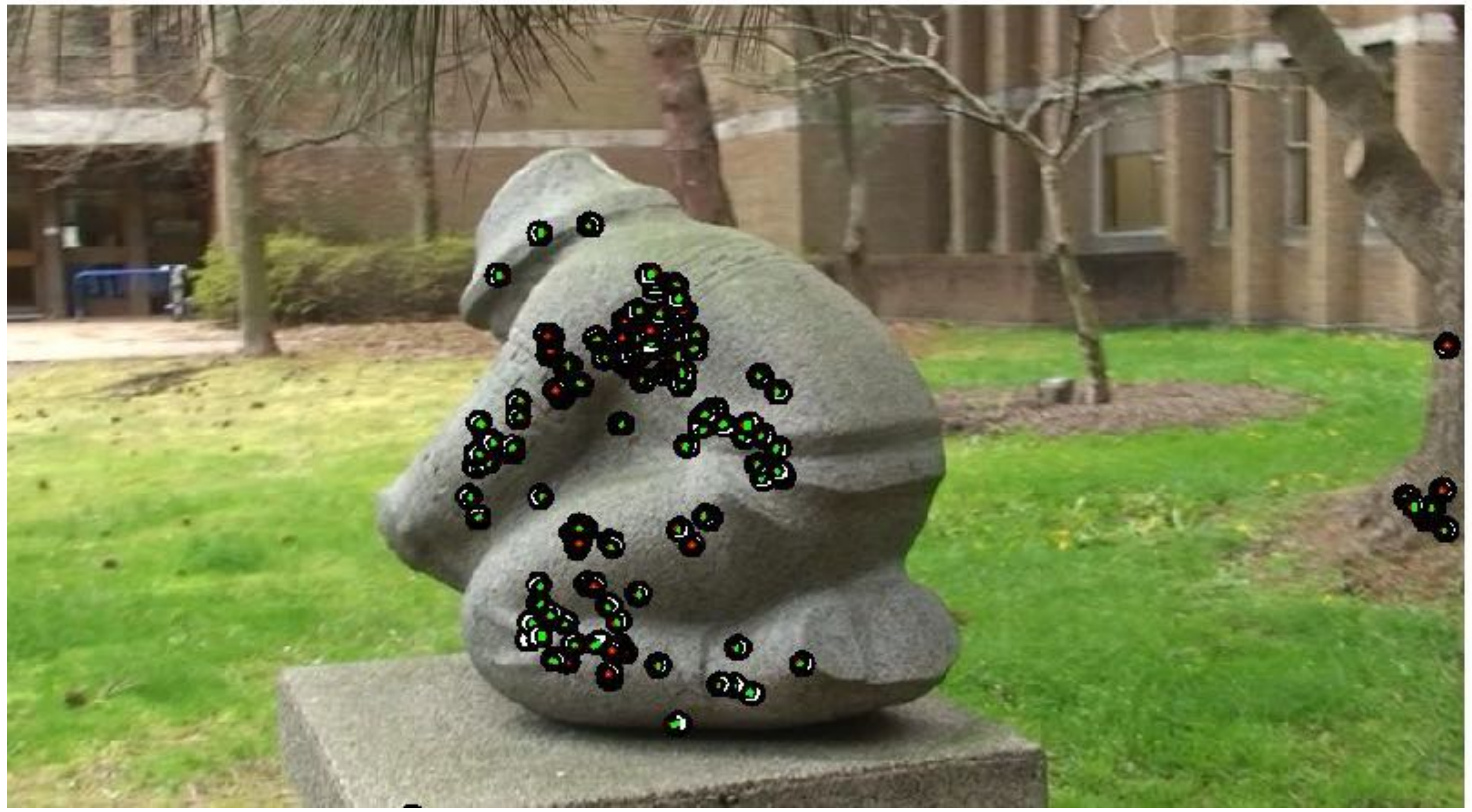


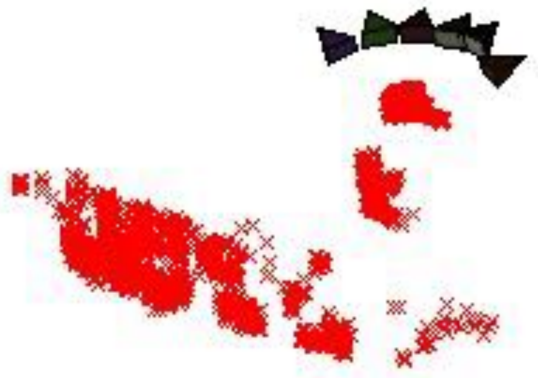


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6

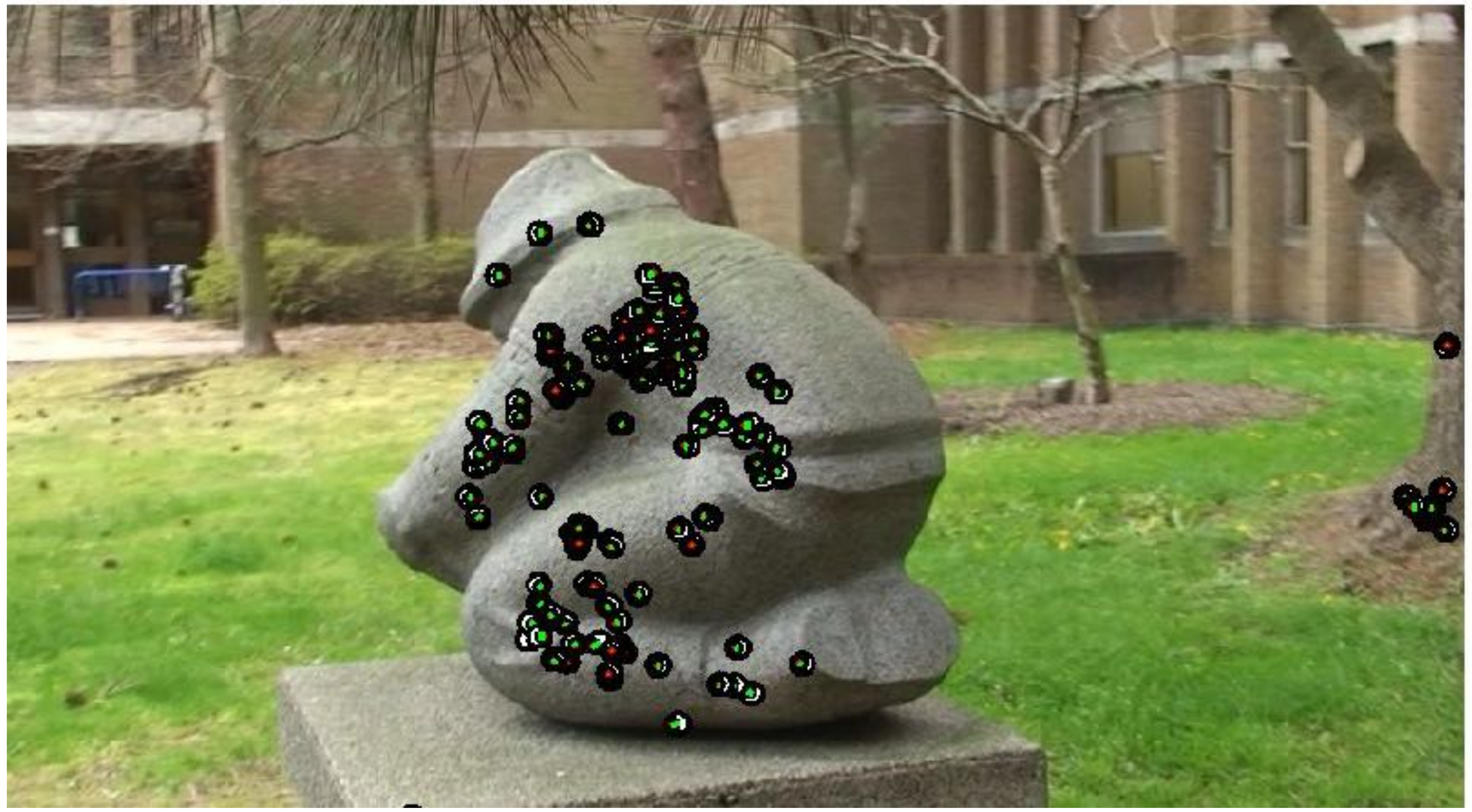


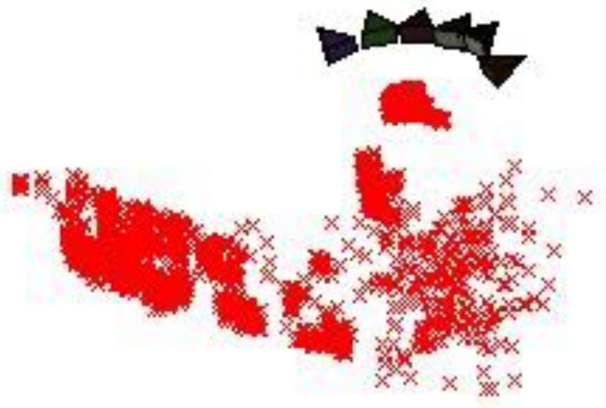


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6

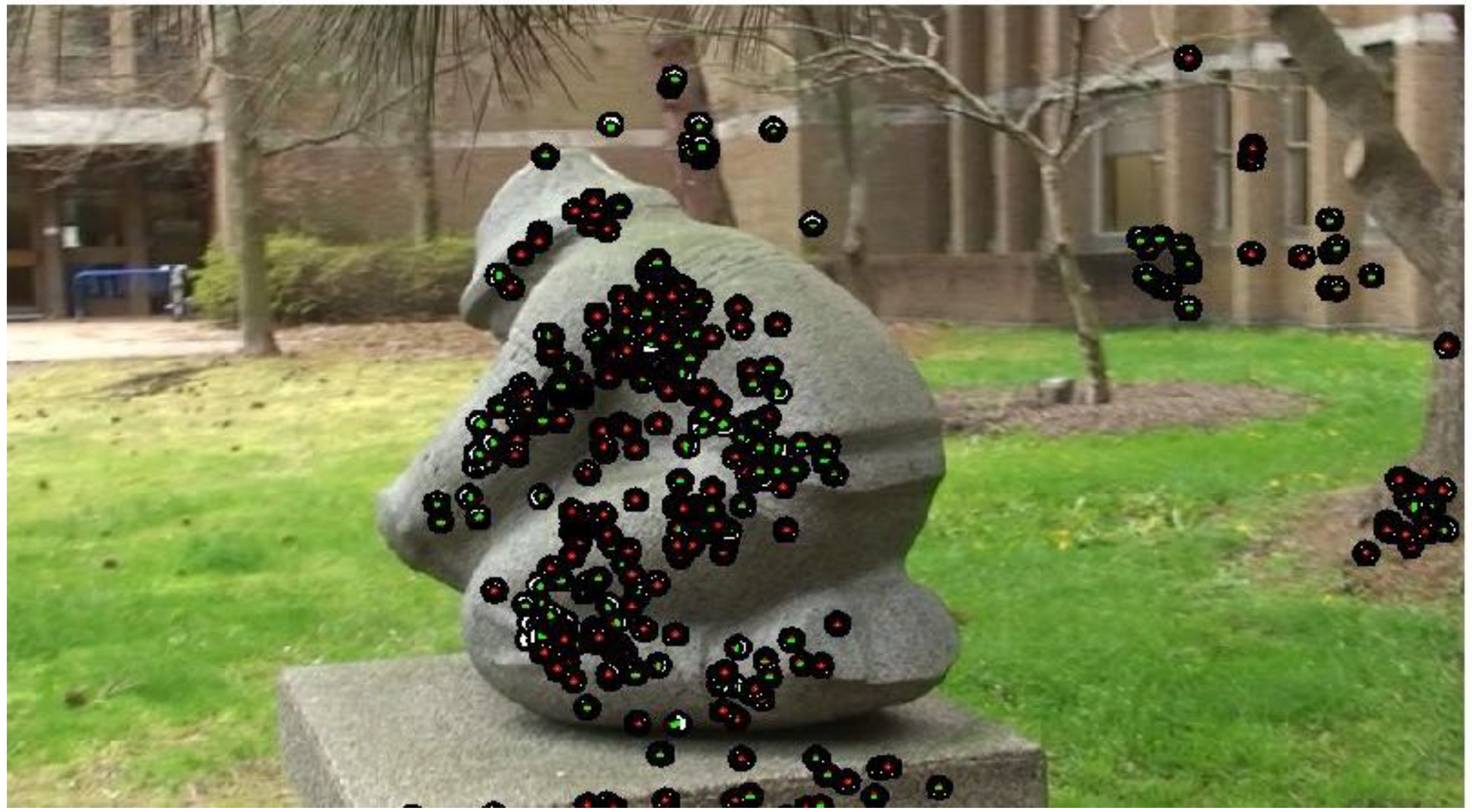


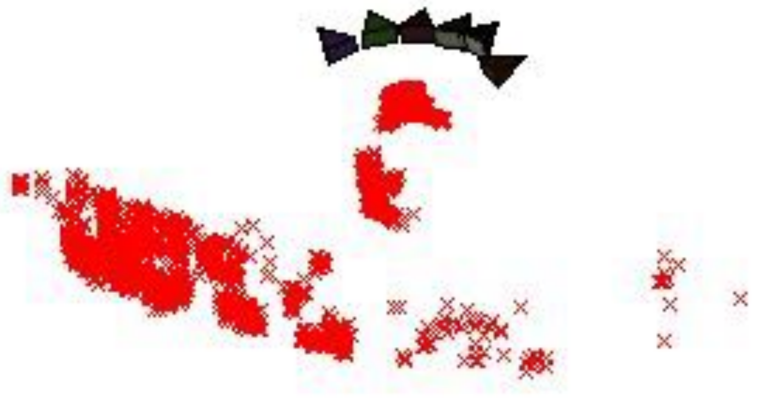


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6

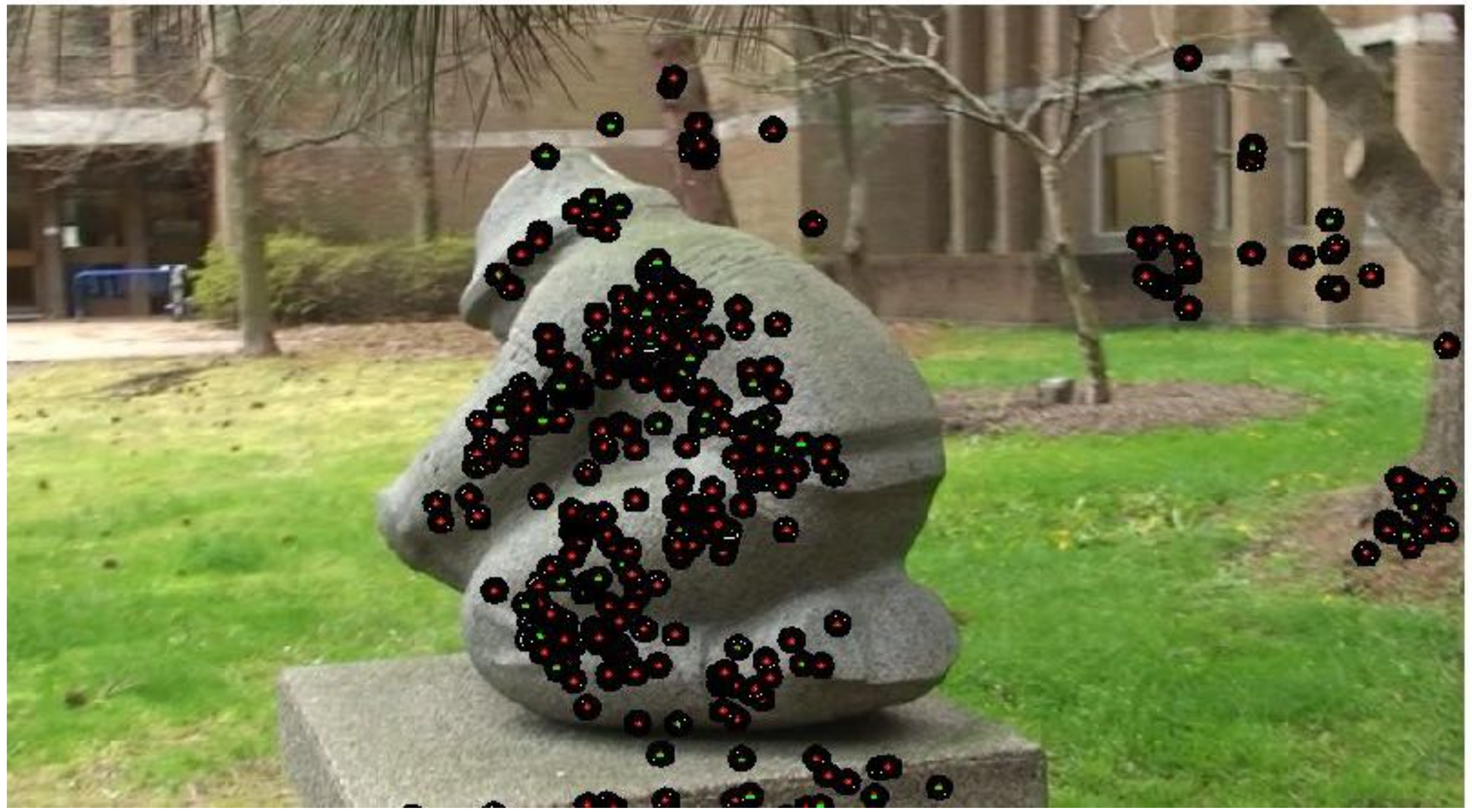


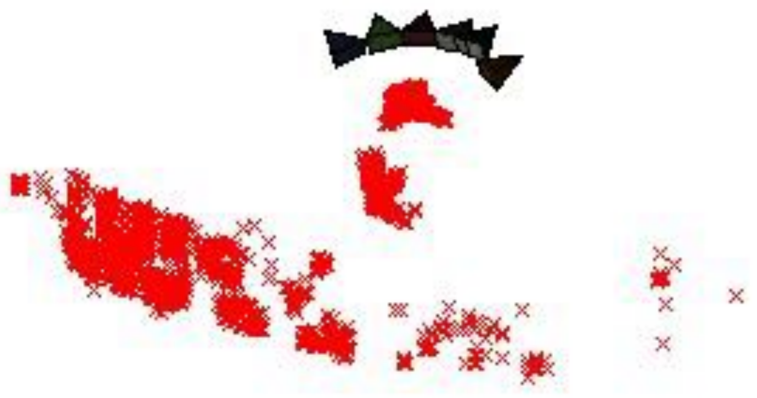


Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

- Add camera 6





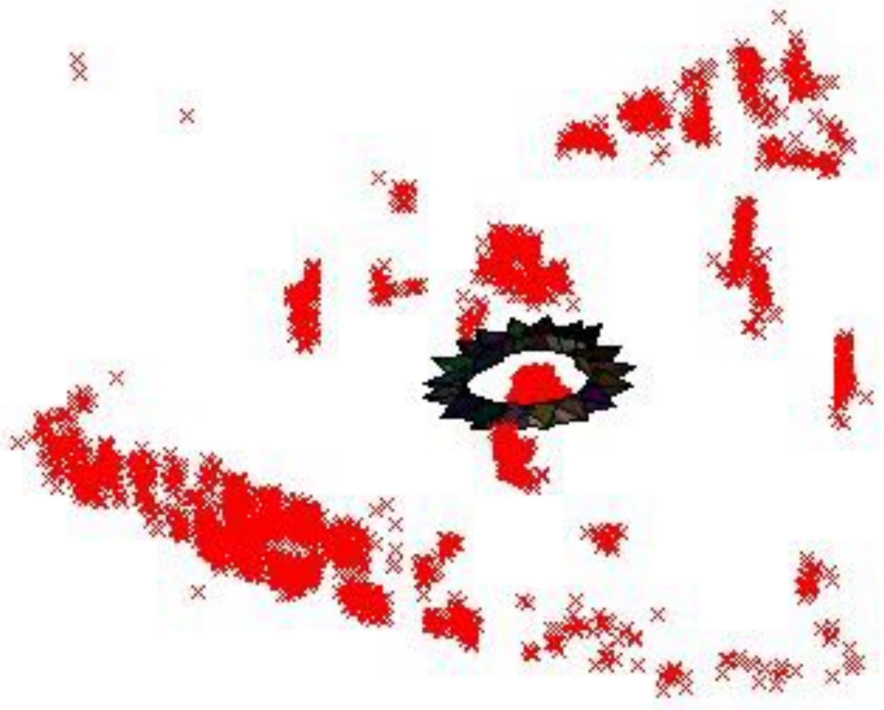
Estimate camera pose, add new 3D points, jointly optimize

Bundle Adjustment

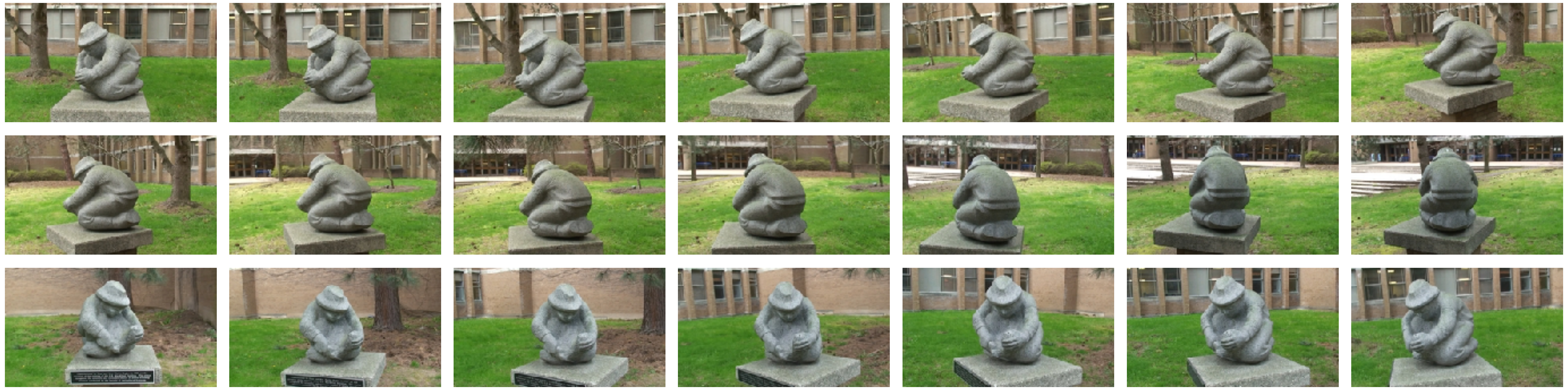
- Add remaining cameras in same way

Bundle Adjustment

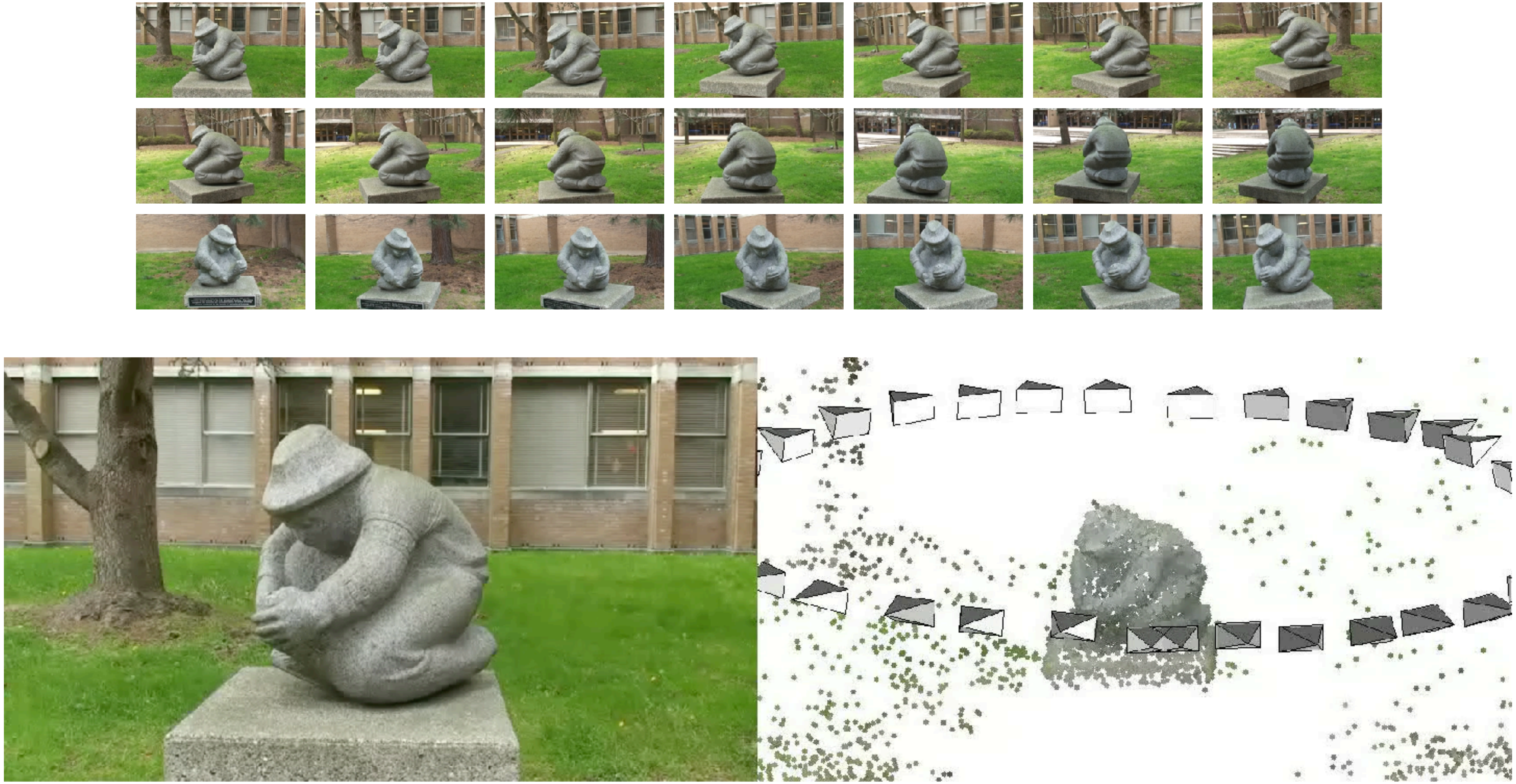
- Add remaining cameras in same way



Structure from Motion



Structure from Motion



SFM recap

SFM recap

- Match features, e.g., SIFT, between all views

SFM recap

- Match features, e.g., SIFT, between all views
- Use RANSAC to reject outliers and estimate Epipolar Geometry / Camera matrices

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

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)

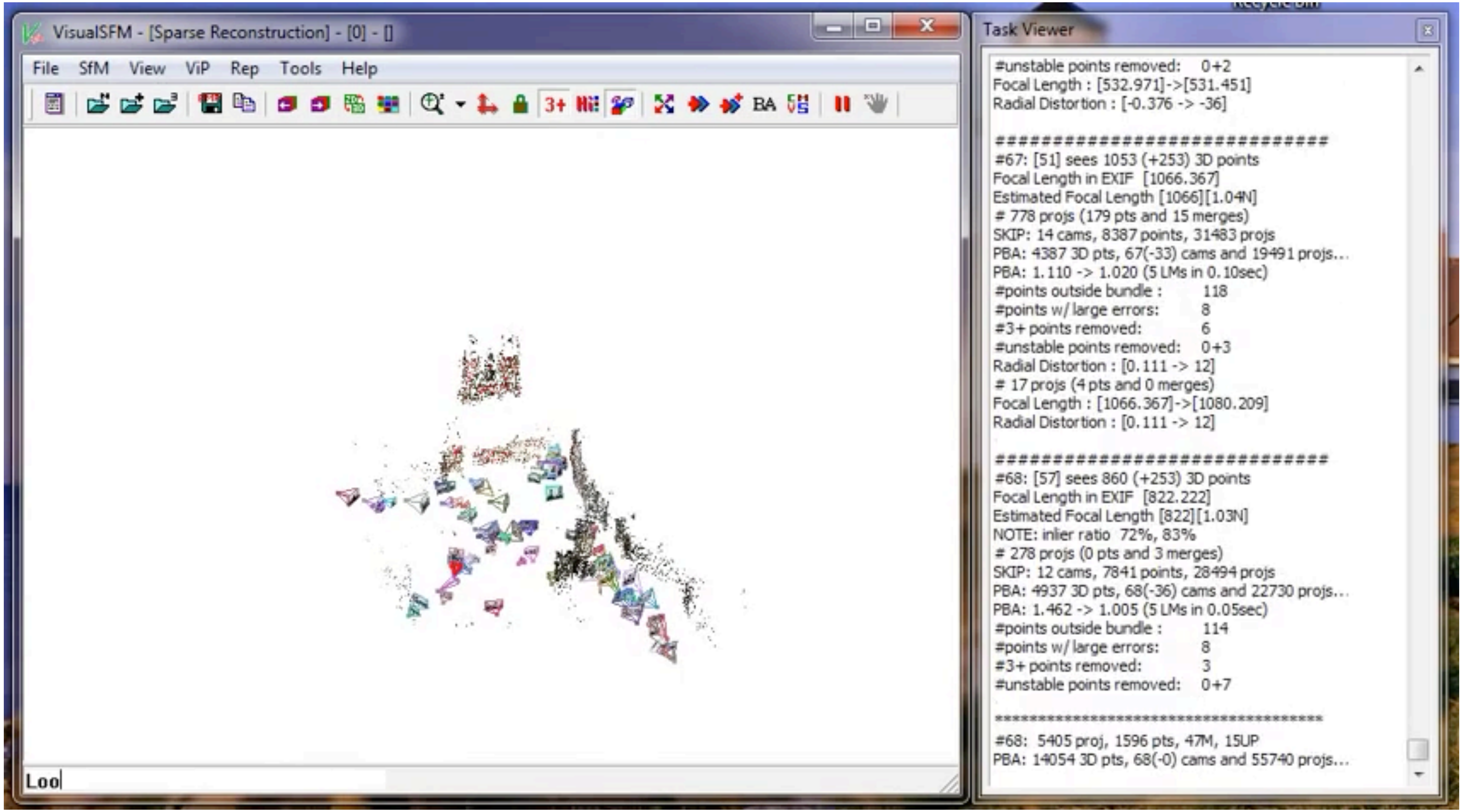
SFM recap

- 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

SFM recap

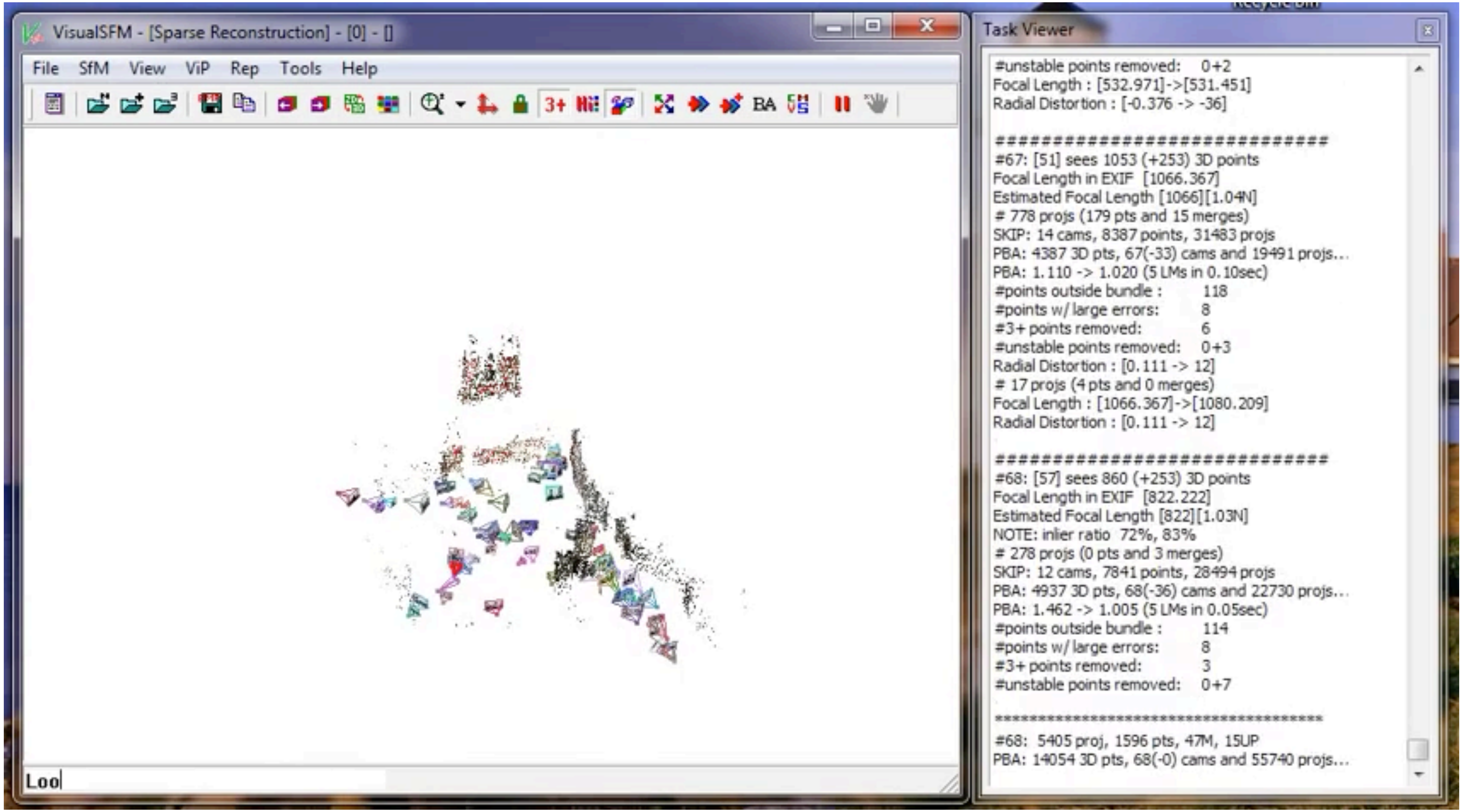
- Match features, e.g., SIFT, between all views
- Use RANSAC to reject outliers and estimate Epipolar Geometry / Camera matrices
- Form feature tracks by linking multiview matches
- Select an initialization set, e.g., 3 images with lots of matches and good baseline (parallax)
- Jointly optimize cameras 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



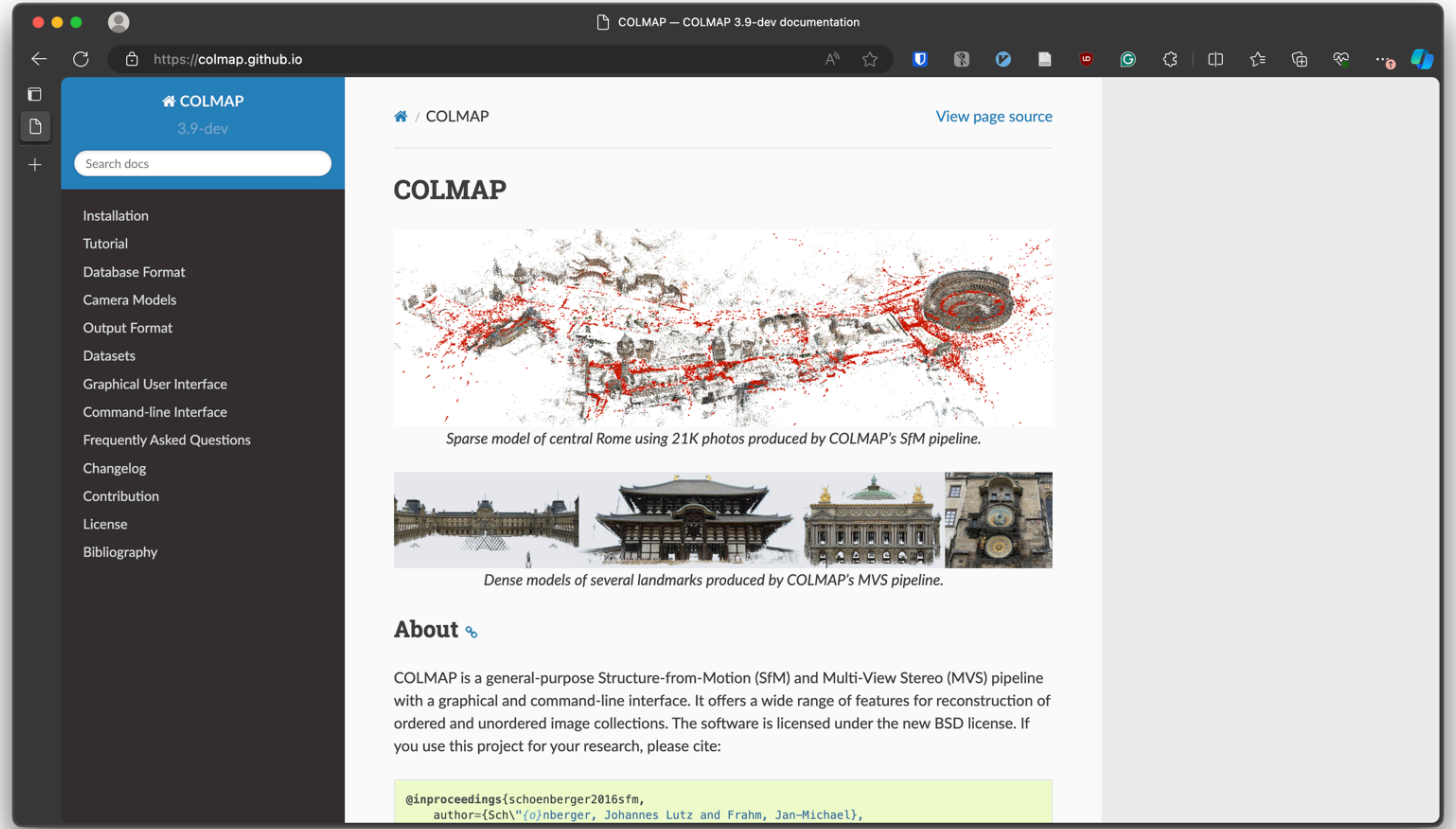
[ccwu.me/vsfm]

Visual SFM



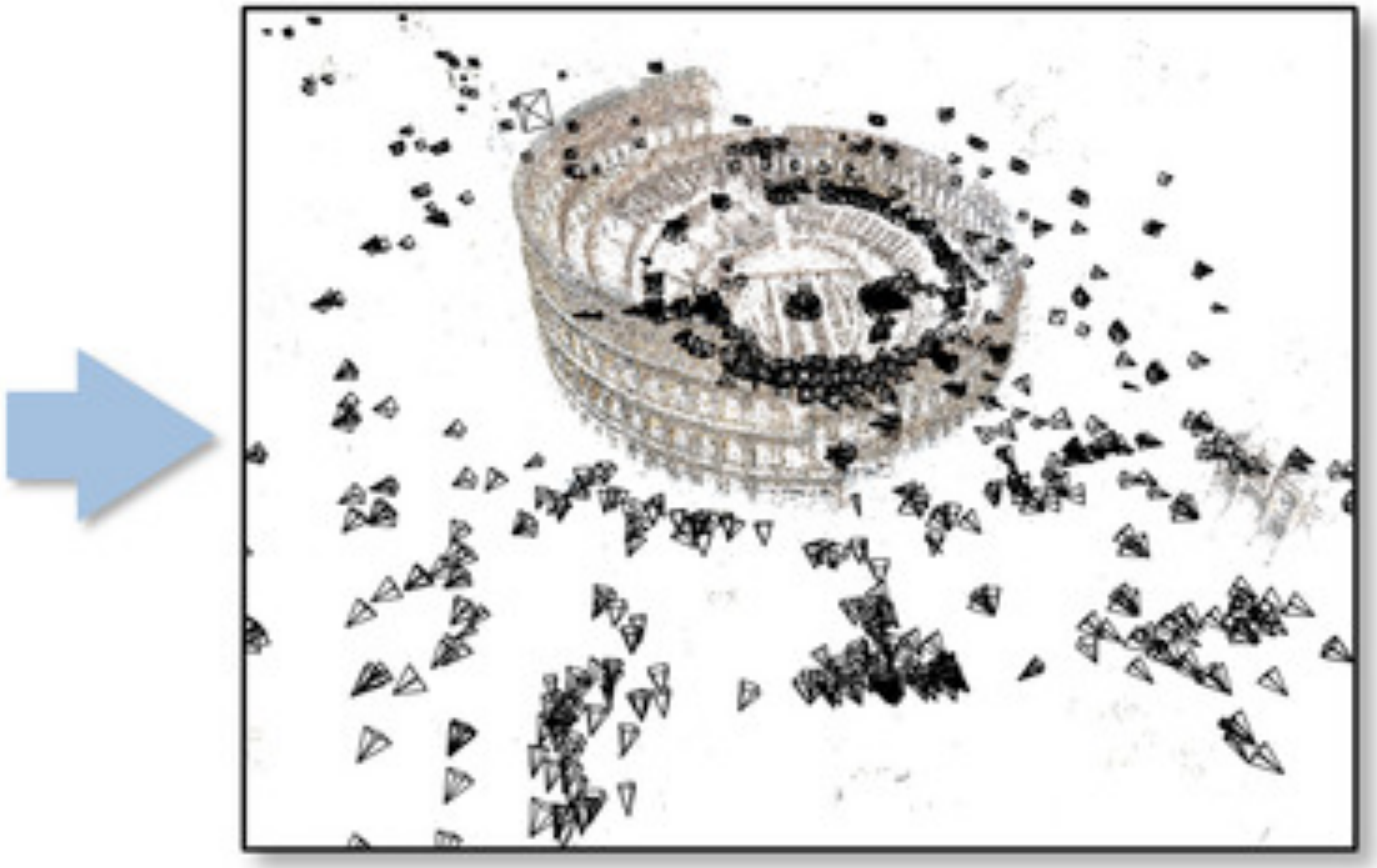
[ccwu.me/vsfm]

COLMAP

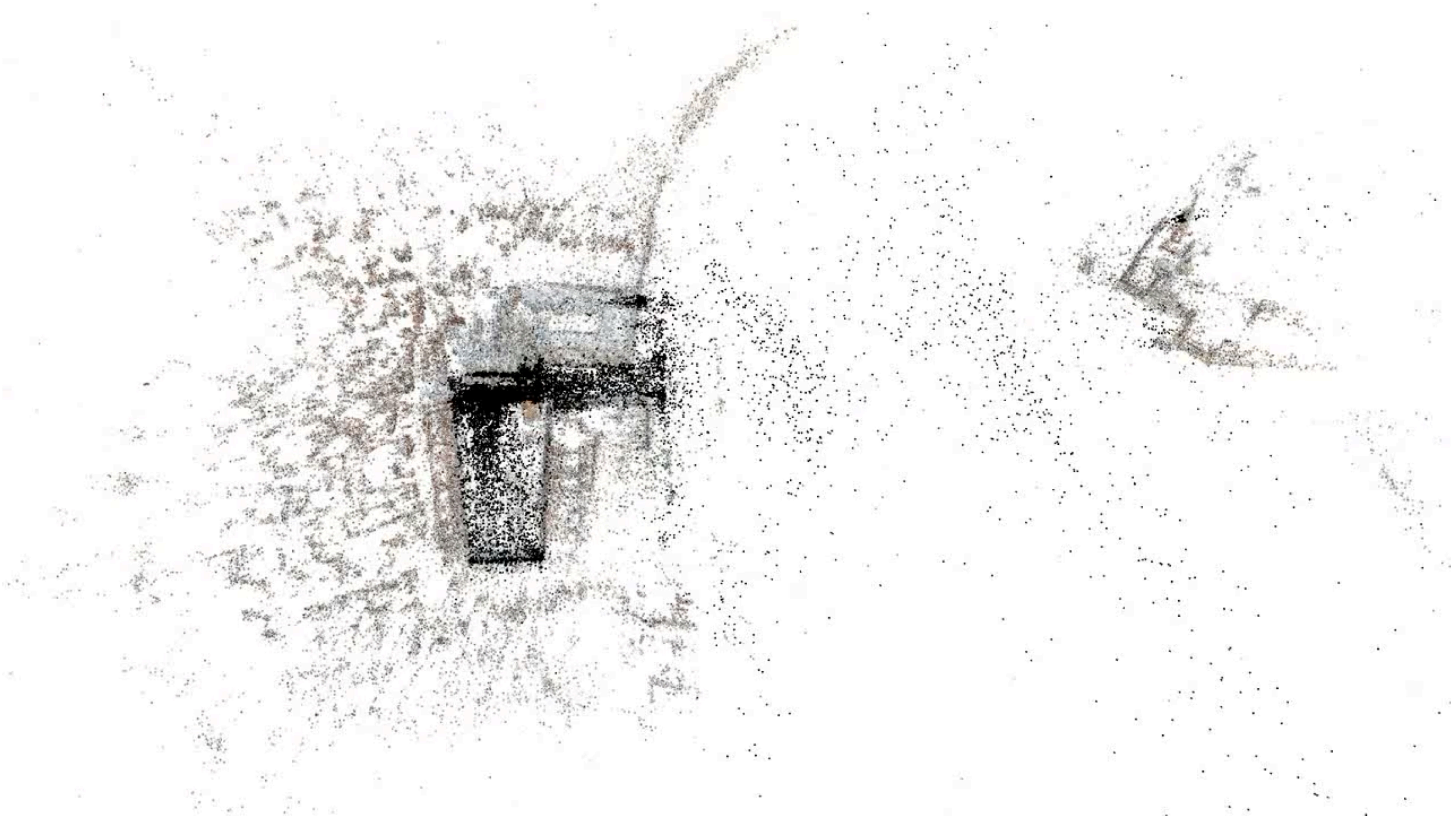

COLMAP — COLMAP 3.9-dev documentation
https://colmap.github.io
COLMAP
3.9-dev
Search docs
Installation
Tutorial
Database Format
Camera Models
Output Format
Datasets
Graphical User Interface
Command-line Interface
Frequently Asked Questions
Changelog
Contribution
License
Bibliography
COLMAP
View page 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

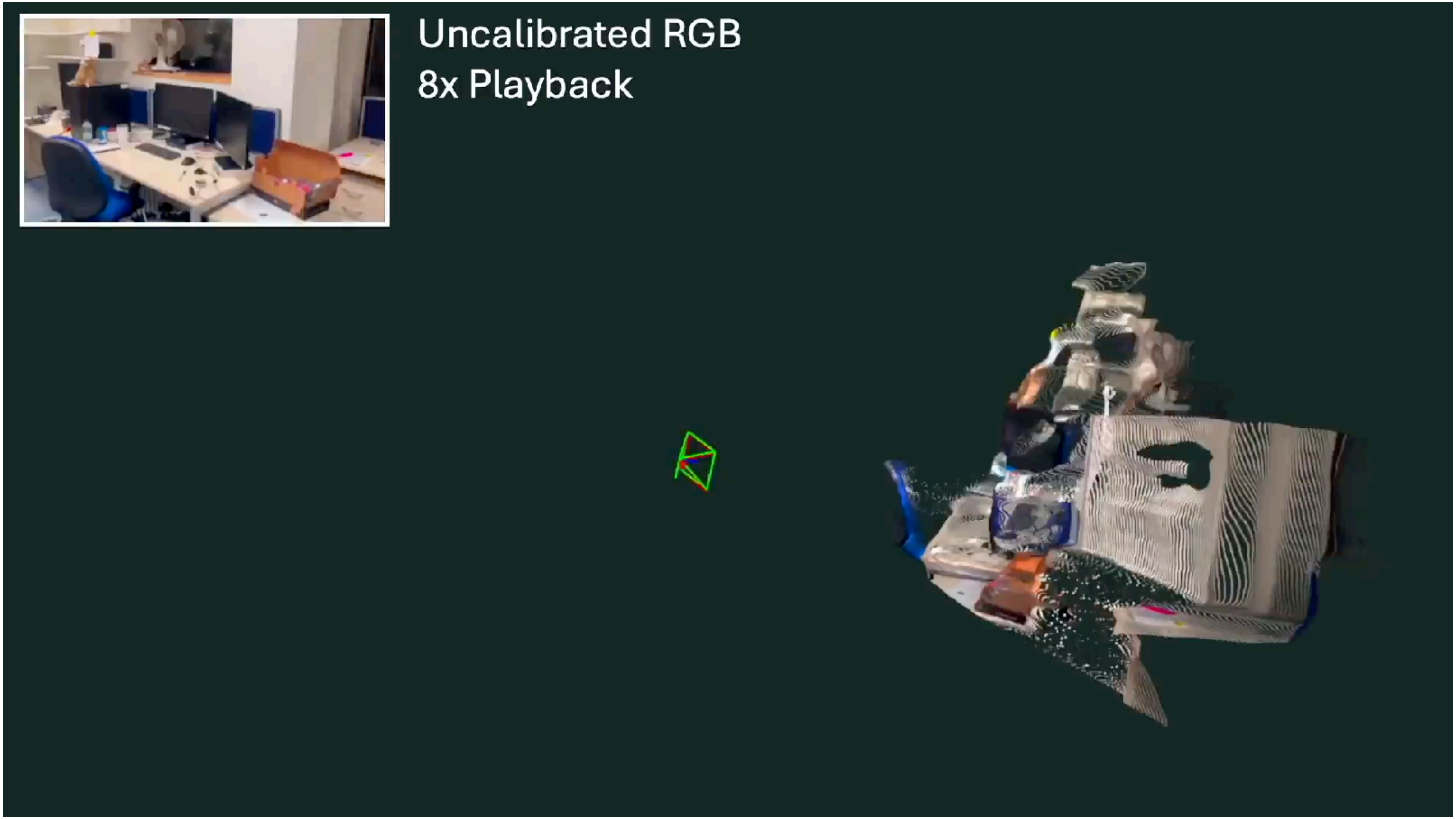
- Reconstruct 3D from unordered photo collections



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

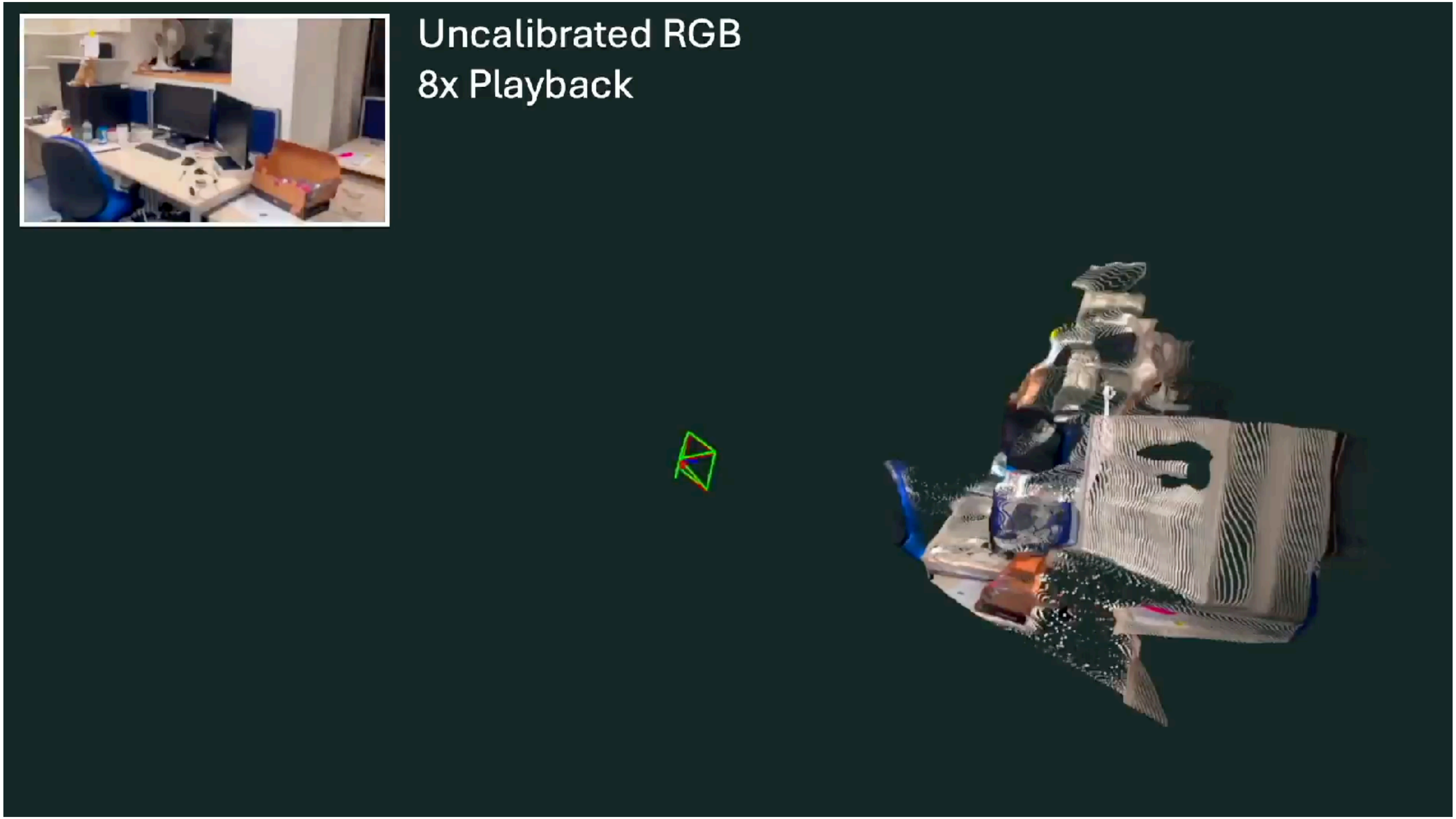


A new paradigm in 2024+



https://edexheim.github.io/mast3r-slam/

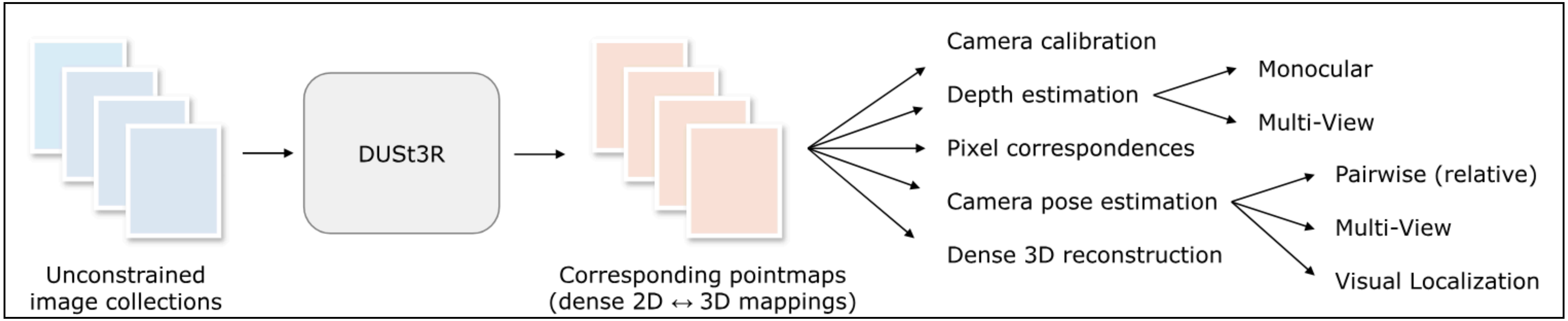
A new paradigm in 2024+



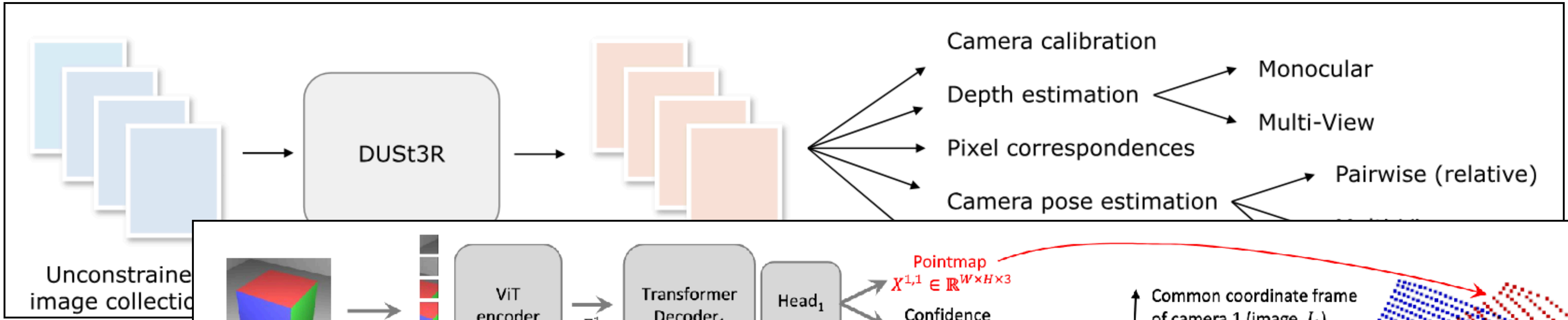
https://edexheim.github.io/mast3r-slam/

A new paradigm in 2024+: “geometry first”

A new paradigm in 2024+: ~~"geometry first"~~ point maps



A new paradigm in 2024+: ~~“geometry first”~~ point maps



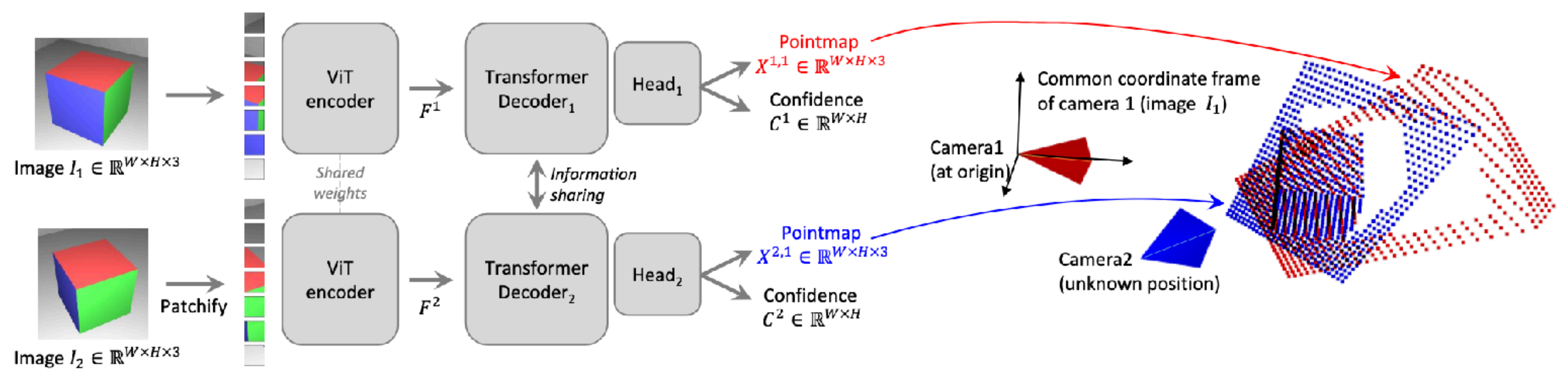
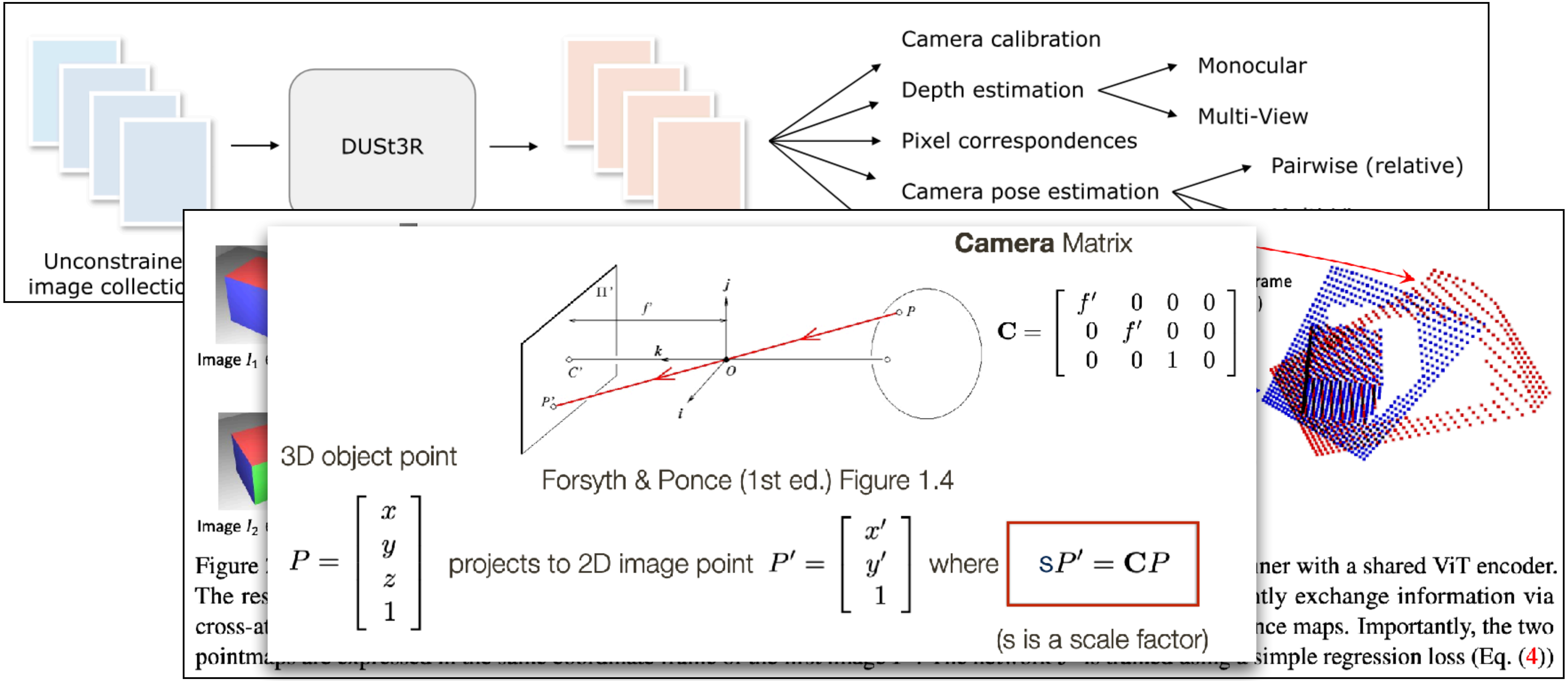


Figure 2. **Architecture of the network $\mathcal{F}$.** Two views of a scene (I^1, I^2) are first encoded in a Siamese manner with a shared ViT encoder. The resulting token representations F^1 and F^2 are then passed to two transformer decoders that constantly exchange information via cross-attention. Finally, two regression heads output the two corresponding pointmaps and associated confidence maps. Importantly, the two pointmaps are expressed in the same coordinate frame of the first image I^1. The network $\mathcal{F}$ is trained using a simple regression loss (Eq. (4))

A new paradigm in 2024+: ~~“geometry first”~~ point maps



SFM recap

SFM recap

- Match features, e.g., SIFT, between all views

SFM recap

- Match features, e.g., SIFT, between all views
- Use RANSAC to reject outliers and estimate Epipolar Geometry / Camera matrices

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

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)

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

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