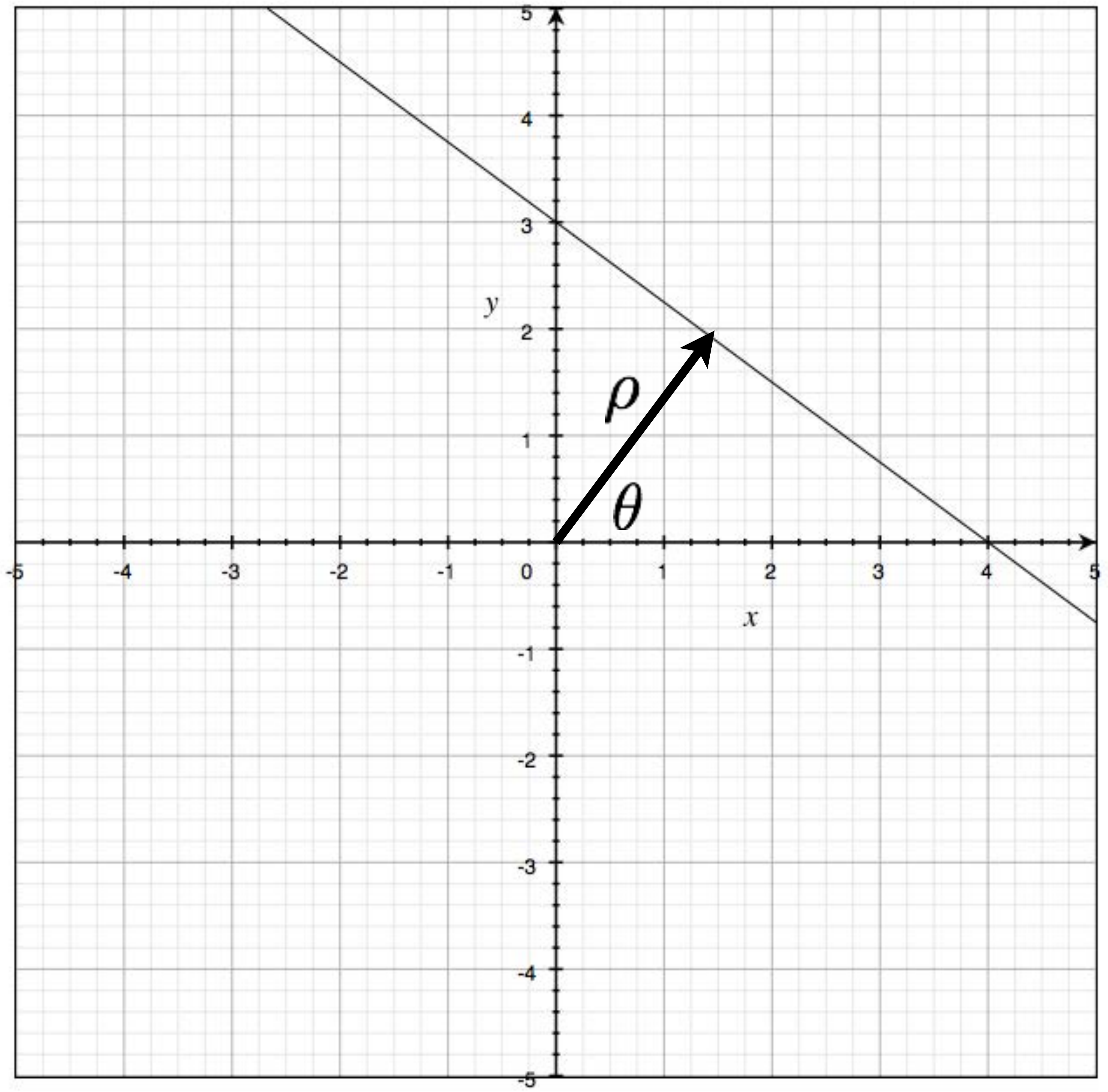
Quiz 4 Feedback

With Bug — Lines: Normal form

$$x \sin\theta + y \cos\theta = \rho$$



Forsyth/Ponce convention

$$x \sin\theta + y \cos\theta + r = 0$$

$$r \geq 0$$

$$0 \leq \theta \leq 2\pi$$

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Bug fixed — Lines: Normal form

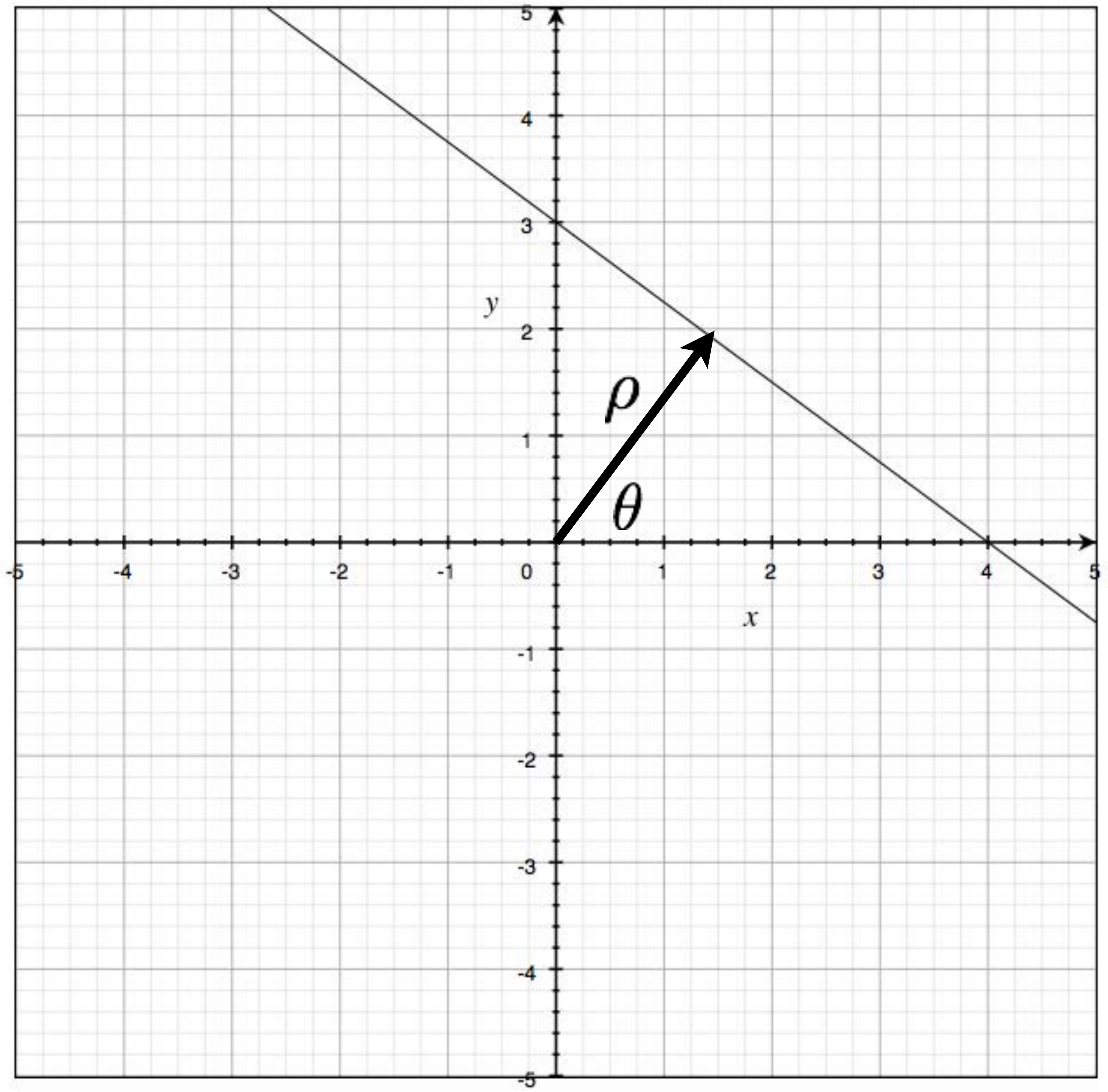
$$x\cos(\theta) + y\sin(\theta) = \rho$$

Forsyth/Ponce convention

$$x\cos(\theta) + y\sin(\theta) + r = 0$$

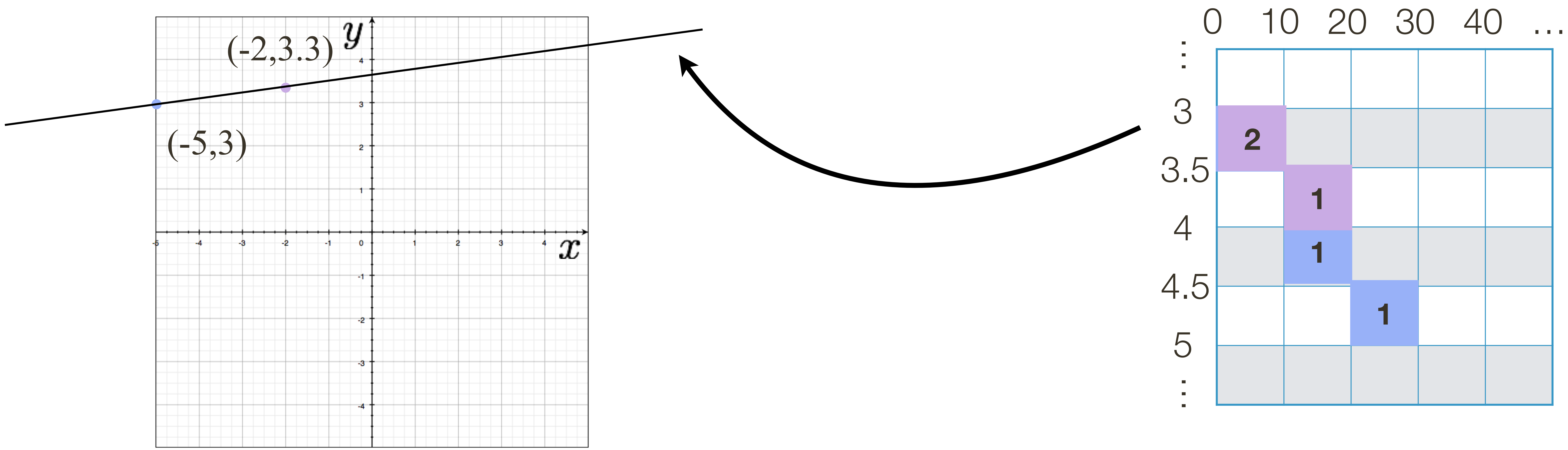
$$r \geq 0$$

$$0 \leq \theta \leq 2\pi$$



Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

With Bug — Example: Hough Transform for Lines



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

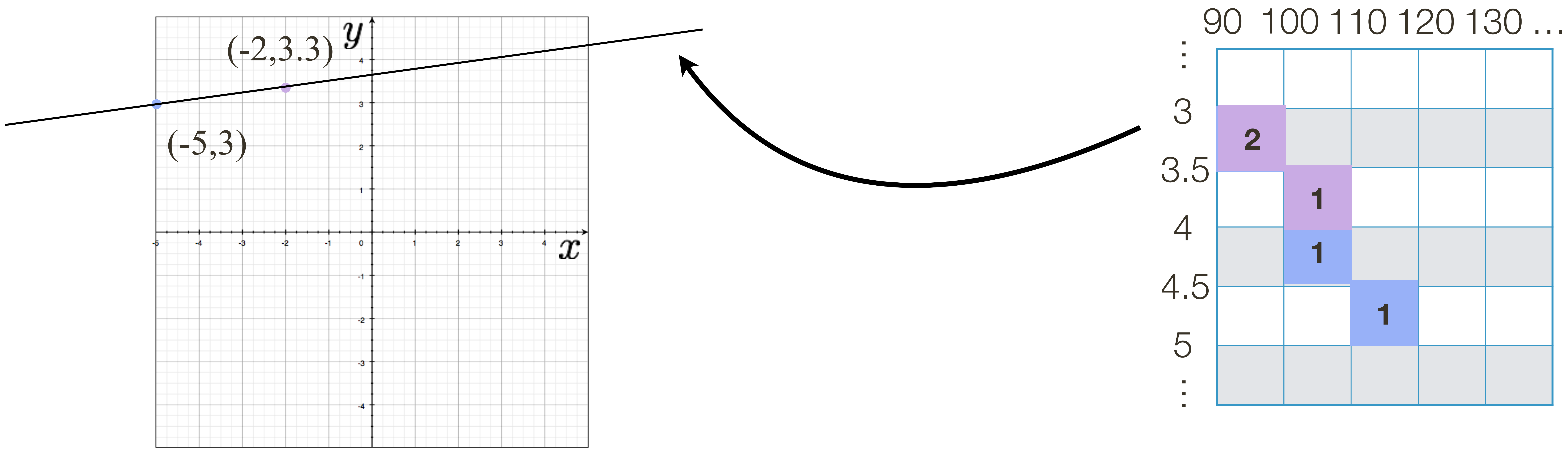
$$-5\sin(15^\circ) - 3\cos(15^\circ) + r = 0 => r = 4.18$$

$$-5\sin(25^\circ) - 3\cos(25^\circ) + r = 0 => r = 4.83$$

$$-2\sin(5^\circ) - 3.3\cos(5^\circ) + r = 0 => r = 3.46$$

$$-2\sin(15^\circ) - 3.3\cos(15^\circ) + r = 0 => r = 3.71$$

Bug fixed — Example: Hough Transform for Lines



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

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

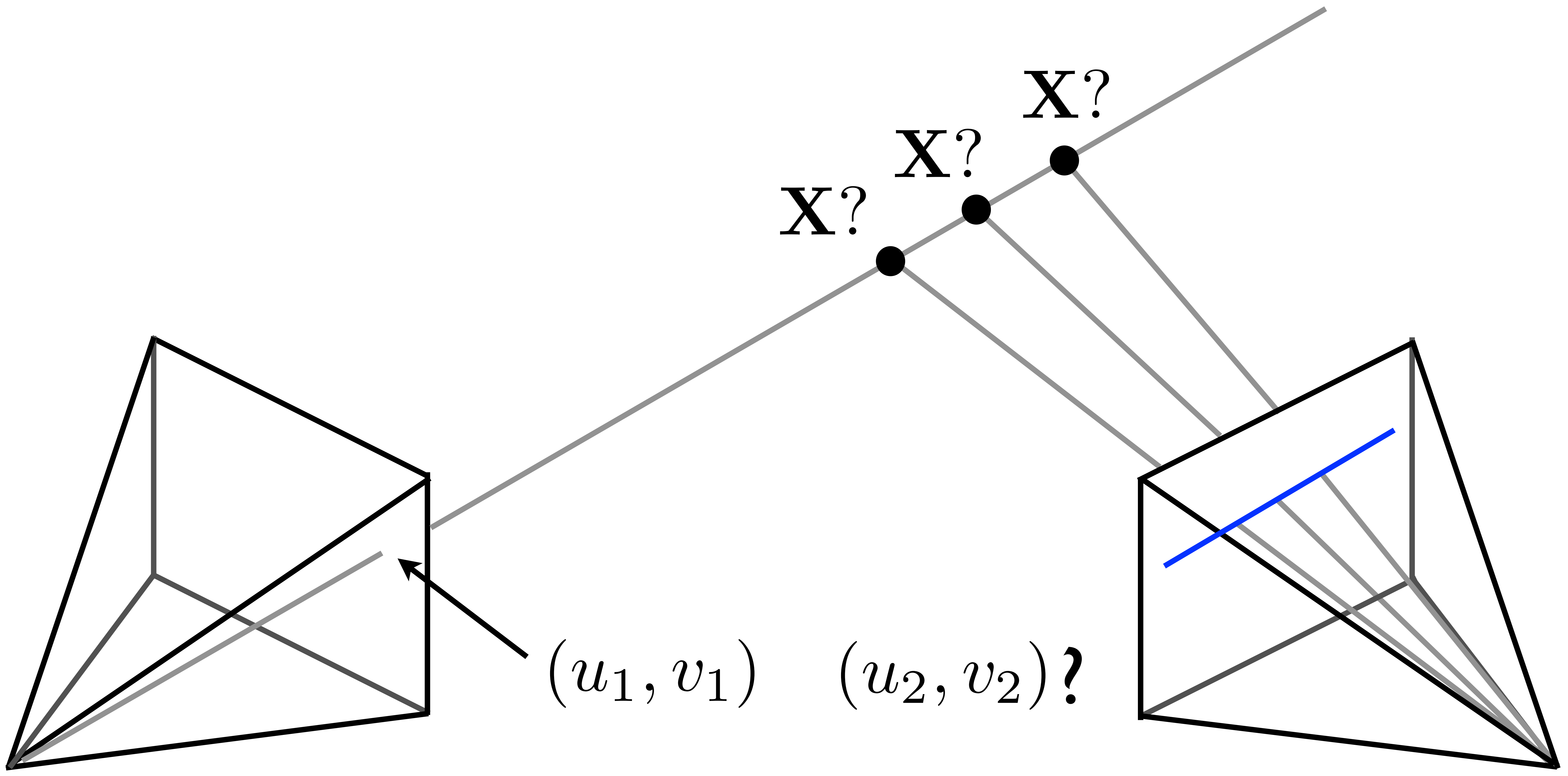
$$-5\cos(115^\circ) + 3\sin(115^\circ) + r = 0 \rightarrow r \approx 4.83$$

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

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

Going back to **Epipolar** Geometry

How do we find correspondences between two views?



A point in Image 1 must lie along the **line** in Image 2

Stereo Matching in Rectified Images

— In a standard stereo setup, where cameras are related by translation in the x direction, epipolar lines are horizontal

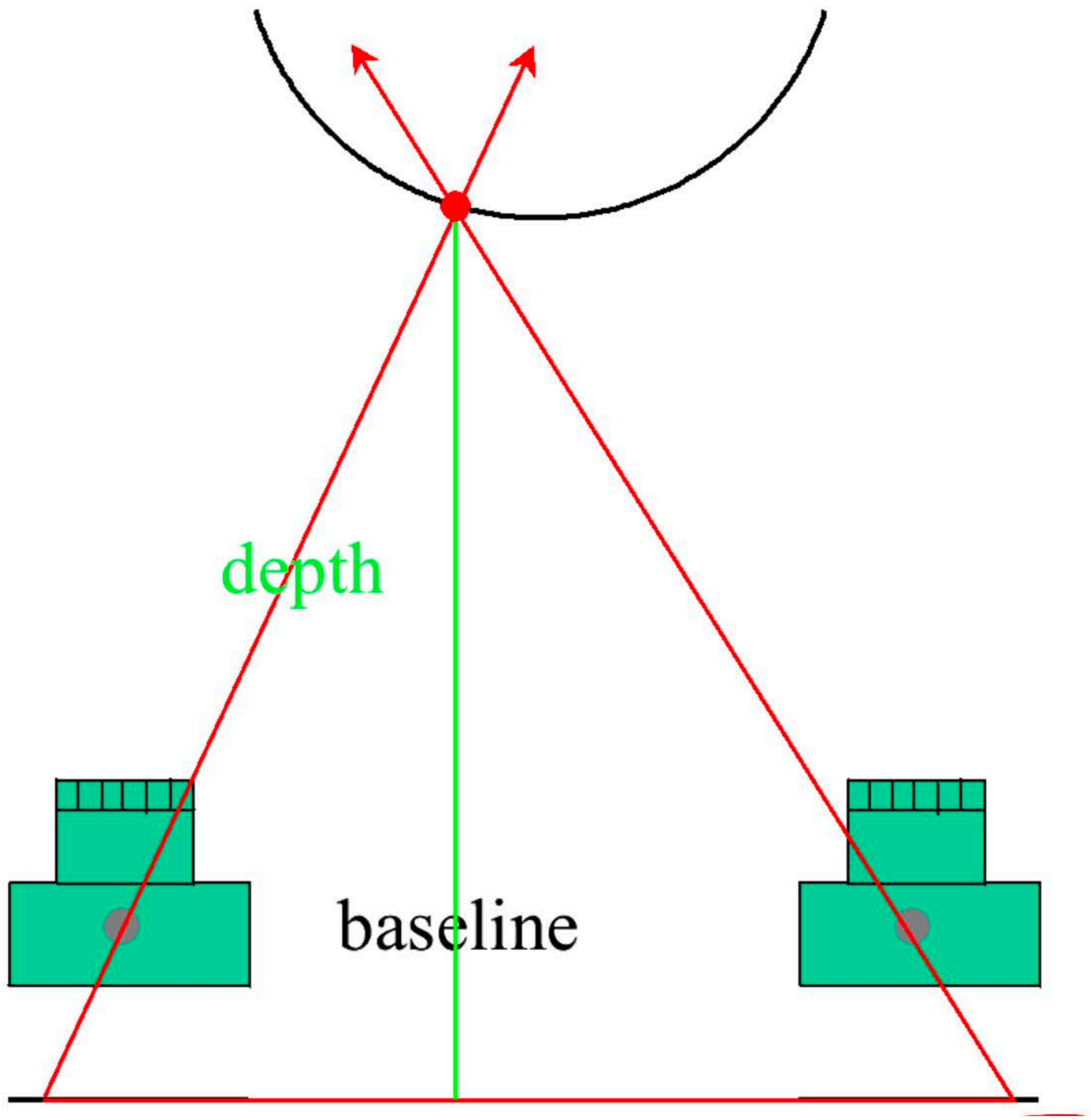
— Stereo algorithms search along scanlines for matches

— Distance along the scanline (difference in x coordinate) for a corresponding feature is called **disparity**

Axis Aligned **Stereo**

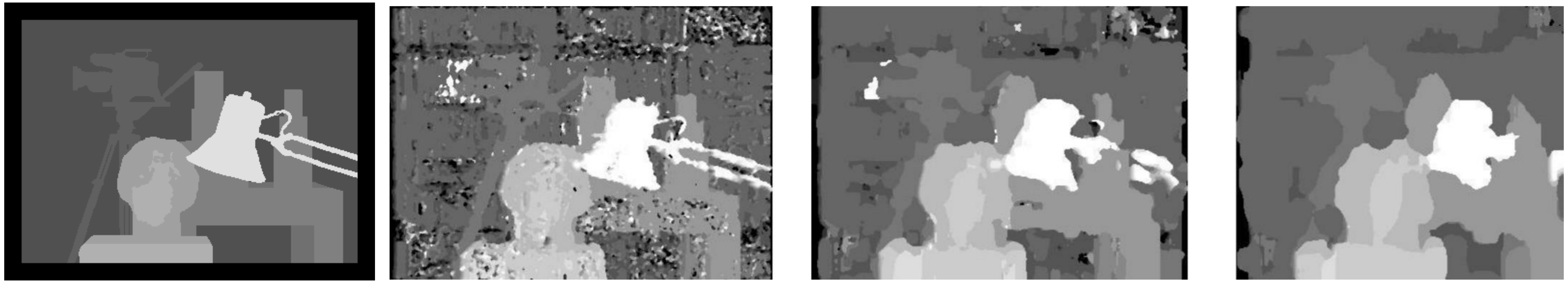
A common stereo configuration has camera optical axes aligned, with cameras related by a translation in the x direction

14.2



Effect of **Window Size**

Larger windows → smoothed result



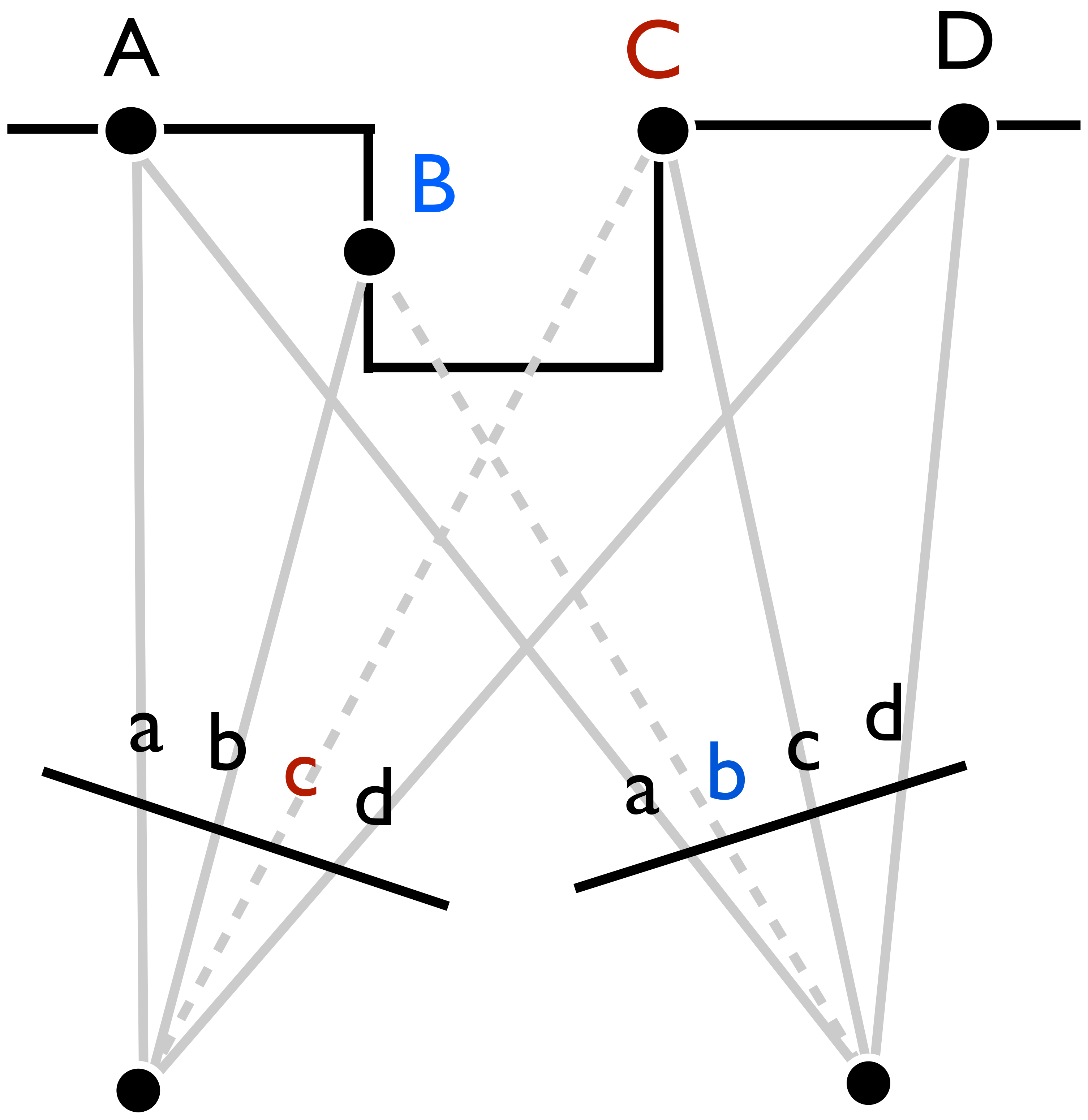
W=3 W=11 W=25

Smaller window
- \+ More detail
- \- More noise

Larger window
- \+ Smoother disparity maps
- \- Less detail
- \- Fails near boundaries

Occlusions

Sometimes a point in image 1 does not appear in image 2, or vice-versa (this is called an **occlusion**)



- Occlusions cause gaps in the stereo reconstruction
- + Matching is difficult nearby as aggregation windows often overlap the occluded region

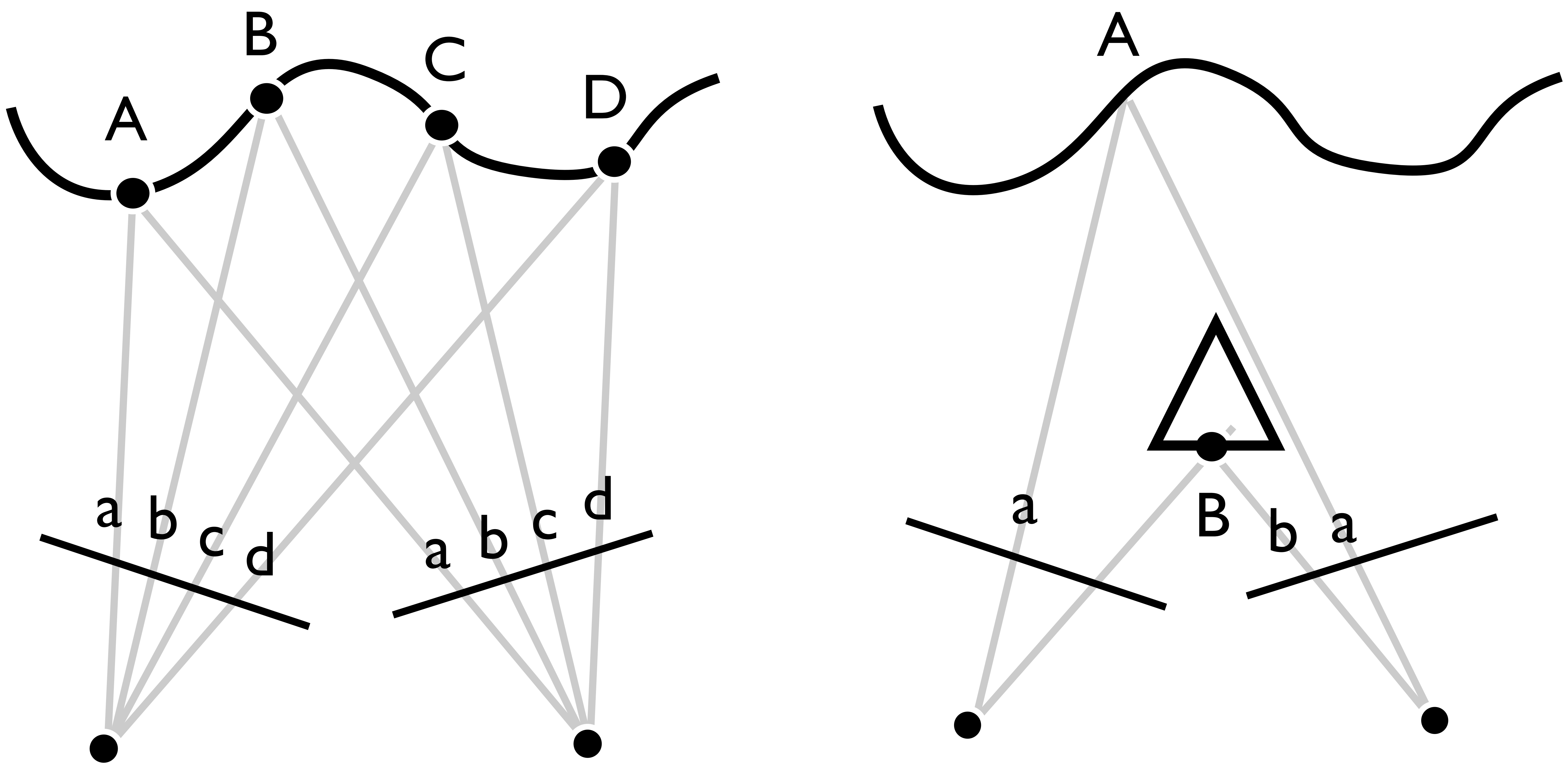
Edge Aware Stereo

Occlusions and depth discontinuities cause problems for stereo matching, as aggregation windows overlap multiple depths

- Segmentation-based stereo approaches aim to solve this by trying to guess the depth edges (e.g., joint segmentation and depth estimation [Taguchi et al 2008])

Ordering Constraint

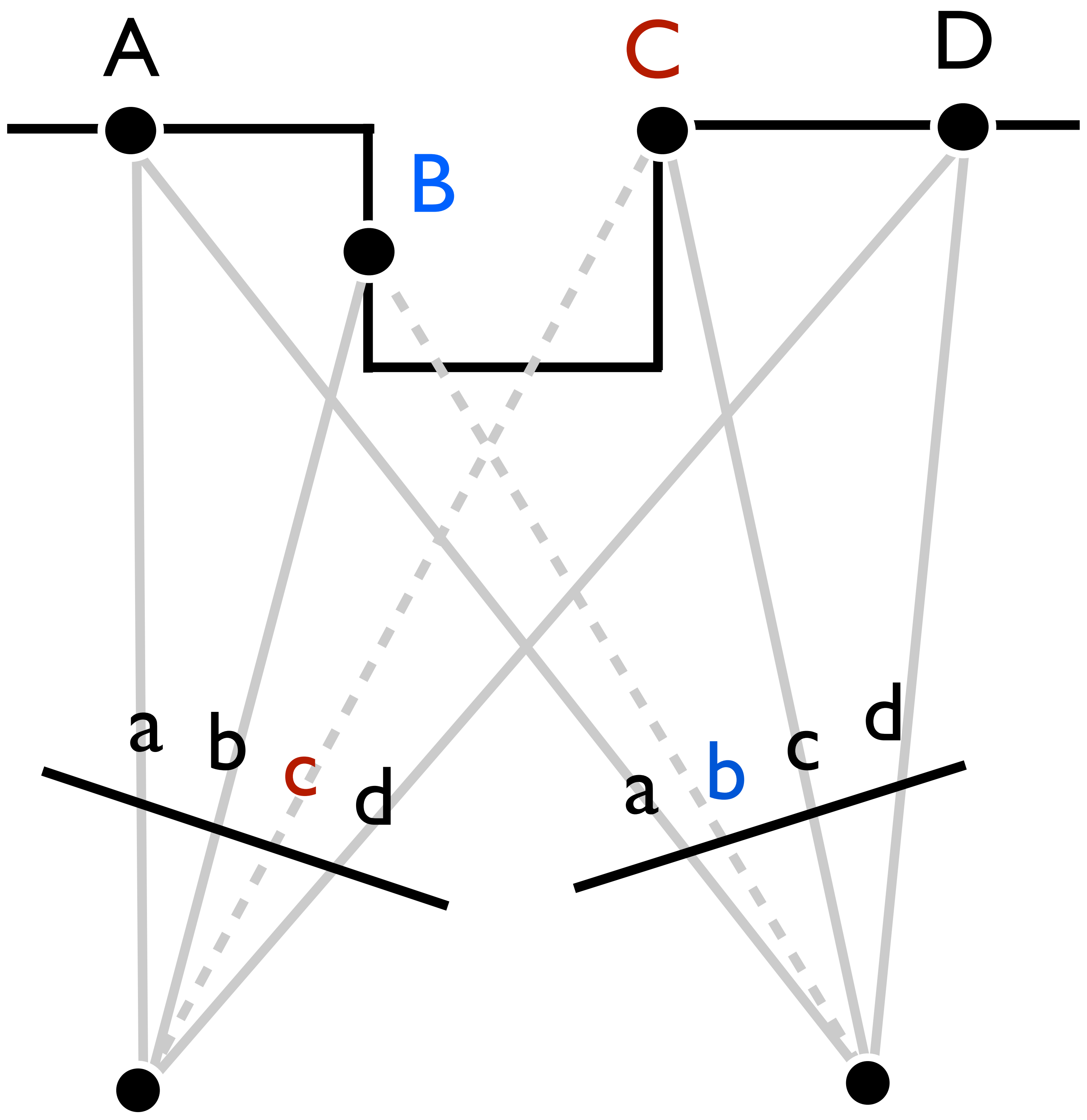
If point B is to the right of point A in image 1, the same is *usually* true in image 2



Not always, e.g., if an object is wholly within the ray triangle generated by A

Occlusions + Ordering

Note that the ordering constraint is still maintained in the presence of occlusions (unless there is an object off surface as in the previous slide)



Stereo Cost Functions

- Energy function for stereo matching based on disparity d(x,y)
- Sum of data and smoothness terms

$$E(d) = E_d(d) + \lambda E_s(d)$$

- Data term is cost of pixel x,y allocated disparity d (e.g., SSD)

$$E_d(d) = \sum_{(x,y)} C(x, y, d(x, y))$$

- Smoothness cost penalises disparity changes with robust ρ(.)

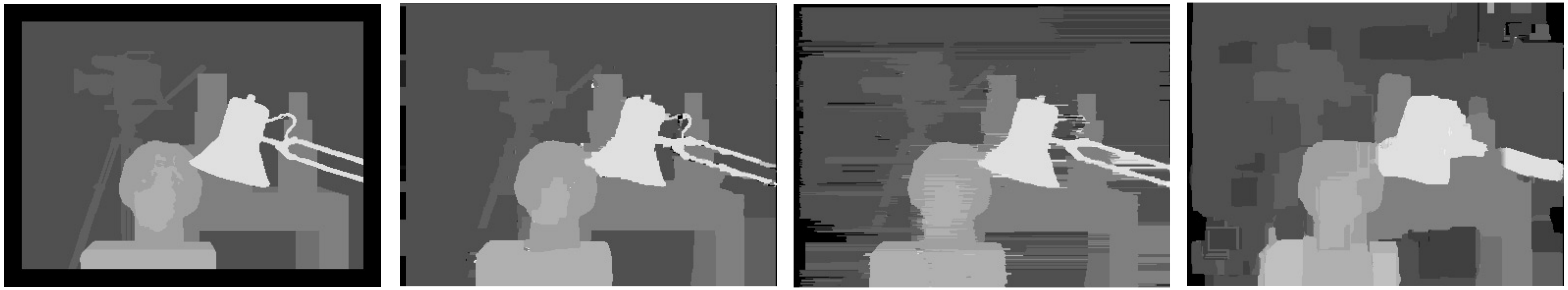
$$E_s(d) = \sum_{(x,y)} \rho(d(x, y) - d(x + 1, y)) + \rho(d(x, y) - d(x, y + 1))$$

- This is a Markov Random Field (MRF), which can be solved using techniques such as Graph Cuts

[Szeliski B5]

Stereo Comparison

- Global vs Scanline vs Local optimization



Ground truth

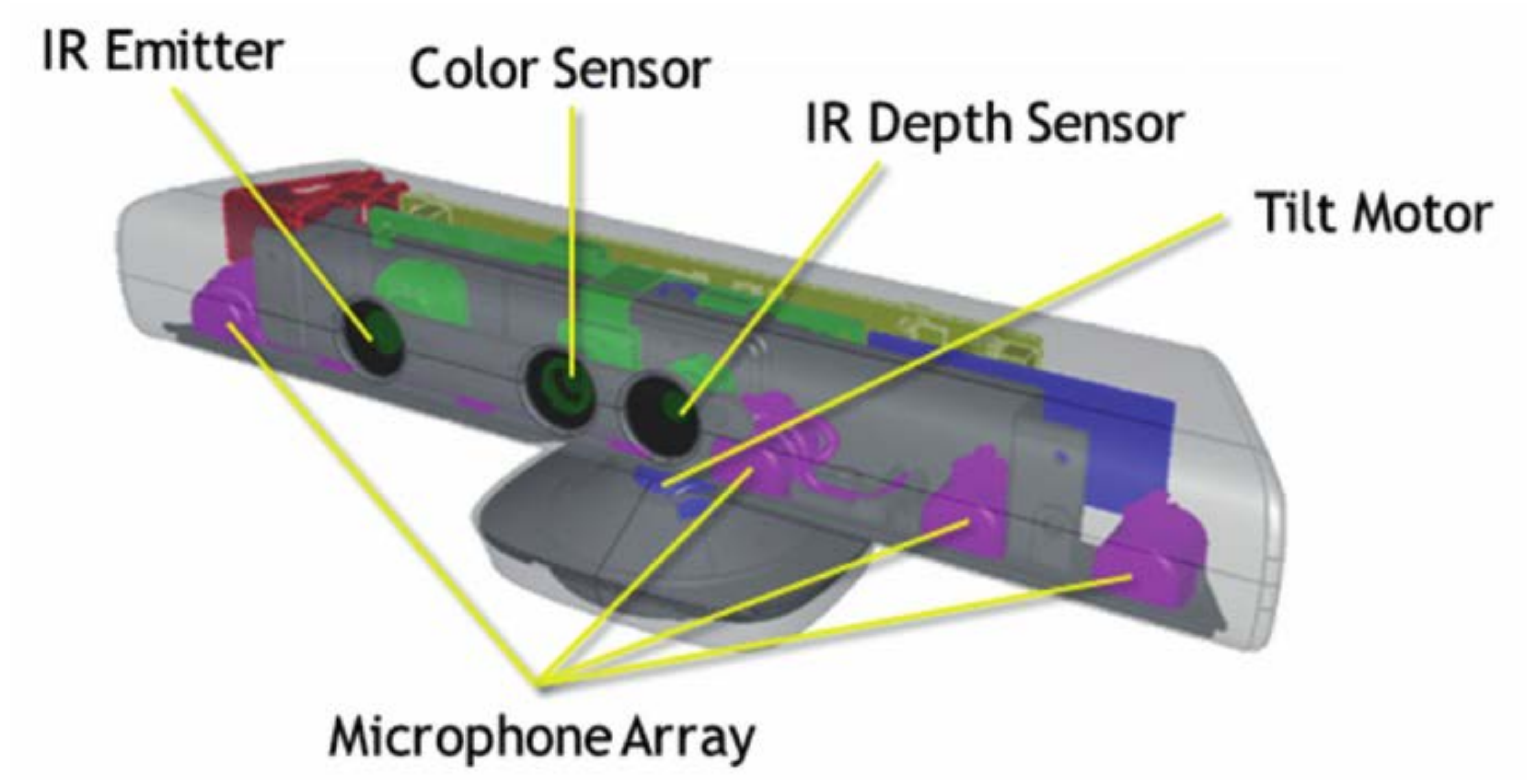
Graph Cuts [Kolmogorov Zabih 2001]

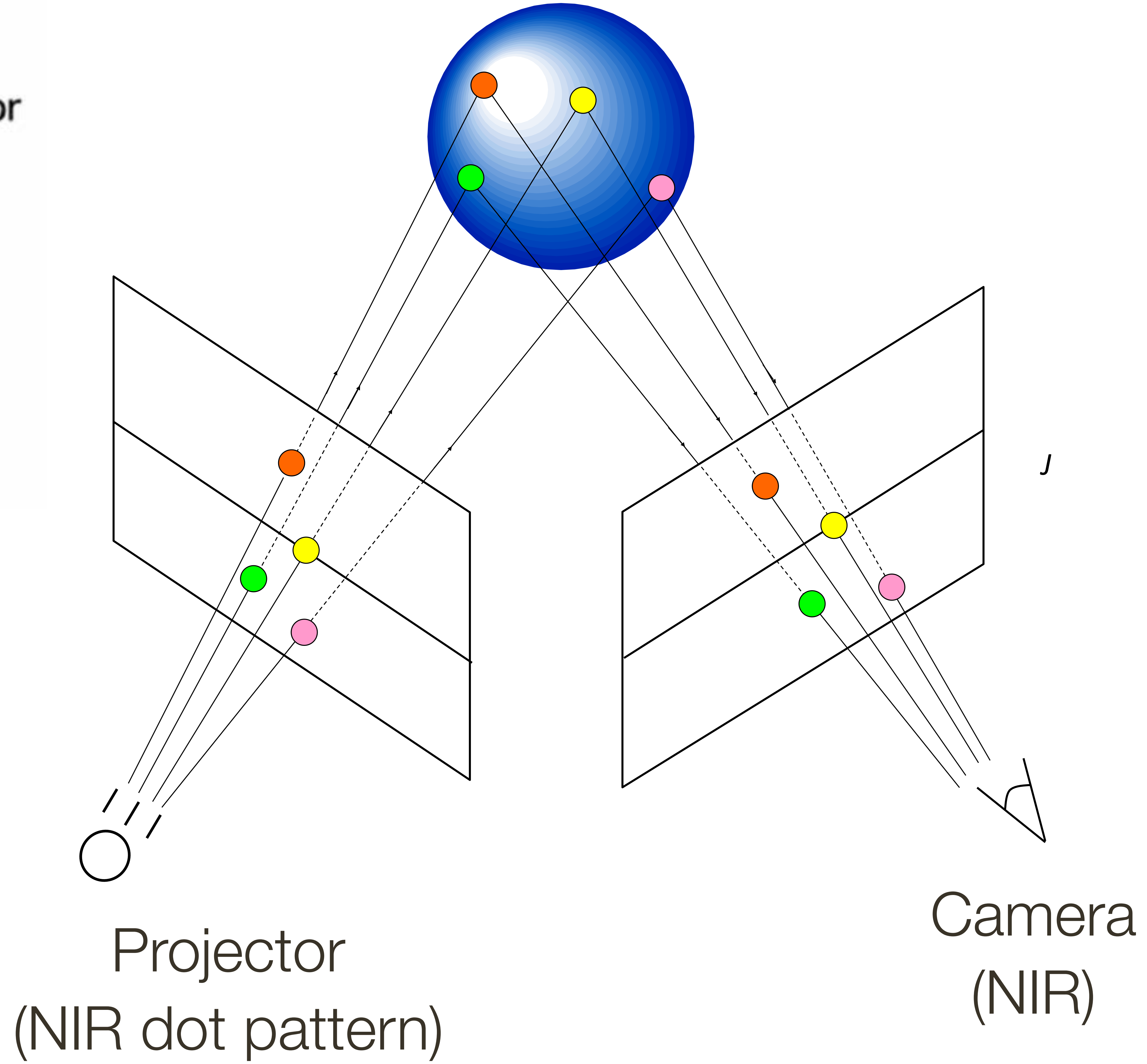
Dynamic Programming

SSD 21px aggregation

[Scharstein Szeliski 2002]

Application: Microsoft **Kinect** v1





Stereo Vision Summary

With two eyes, we acquire images of the world from slightly different viewpoints

We perceive **depth** based on **differences in the relative position of points** in the left image and in the right image

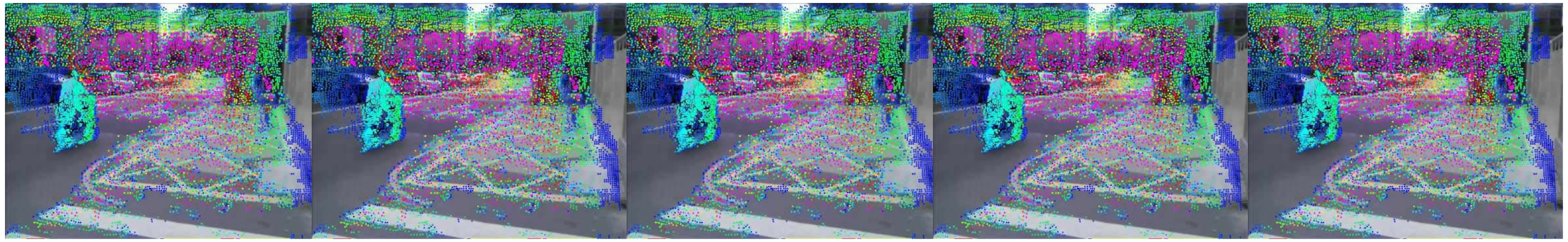
Stereo algorithms work by finding **matches** between points along corresponding lines in a second image, known as epipolar lines.

A point in one image projects to an **epipolar line** in a second image

In an axis-aligned / rectified stereo setup, matches are found along horizontal scanlines

THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 16: Optical Flow

Menu for Today

Topics:

- **Stereo** recap, 1D vs 2D motion
- **Optical Flow**
- **Brightness** Constancy
- **Lucas Kanade**

Readings:

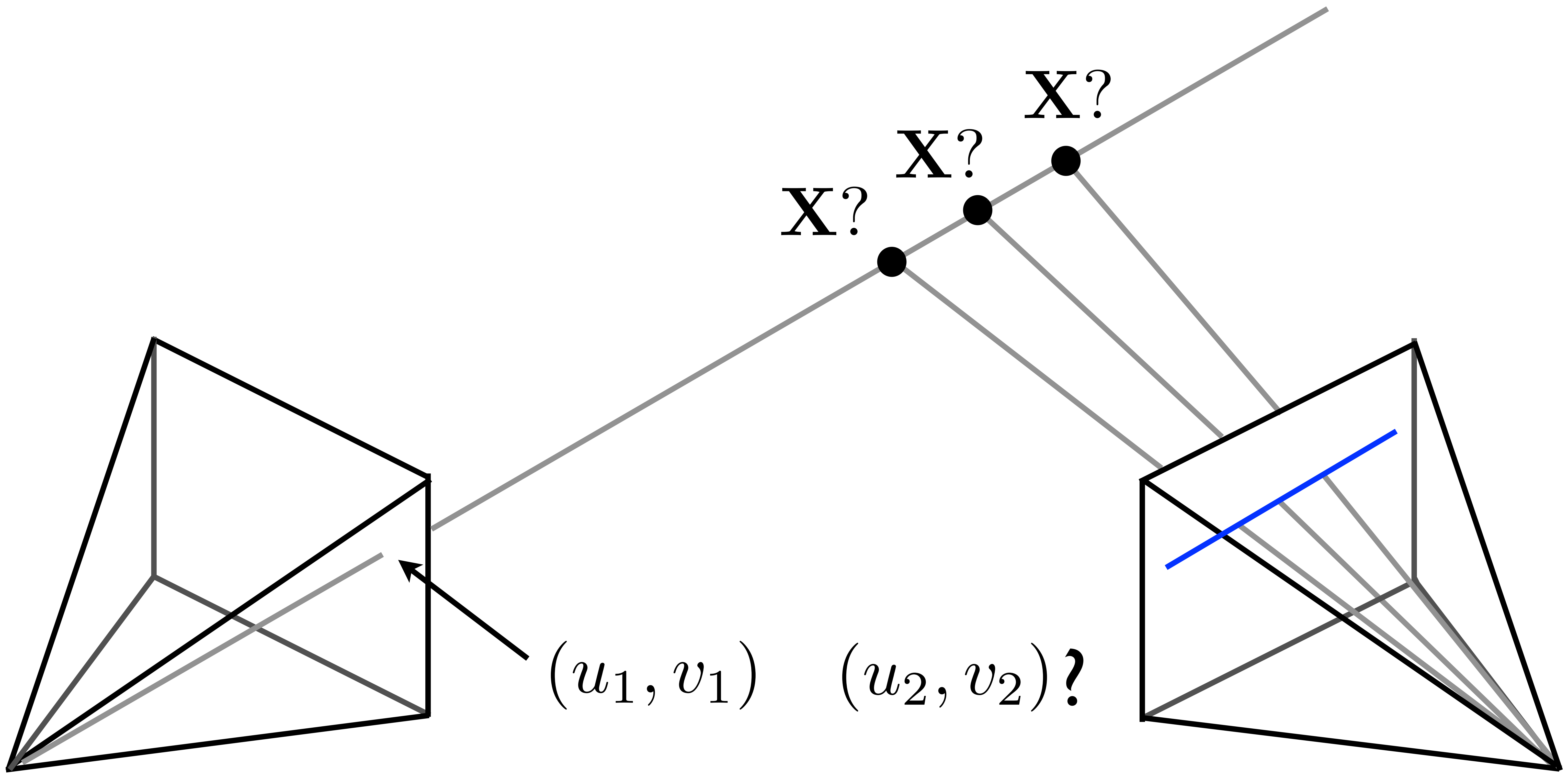
- **Today's** Lecture: Szeliski 12.1, 12.3-12.4, 9.3

Reminders:

- **Assignment 4**: RANSAC and Panoramas due **March 20th**

Epipolar Line

How do we transfer points between 2 views? (non-planar)



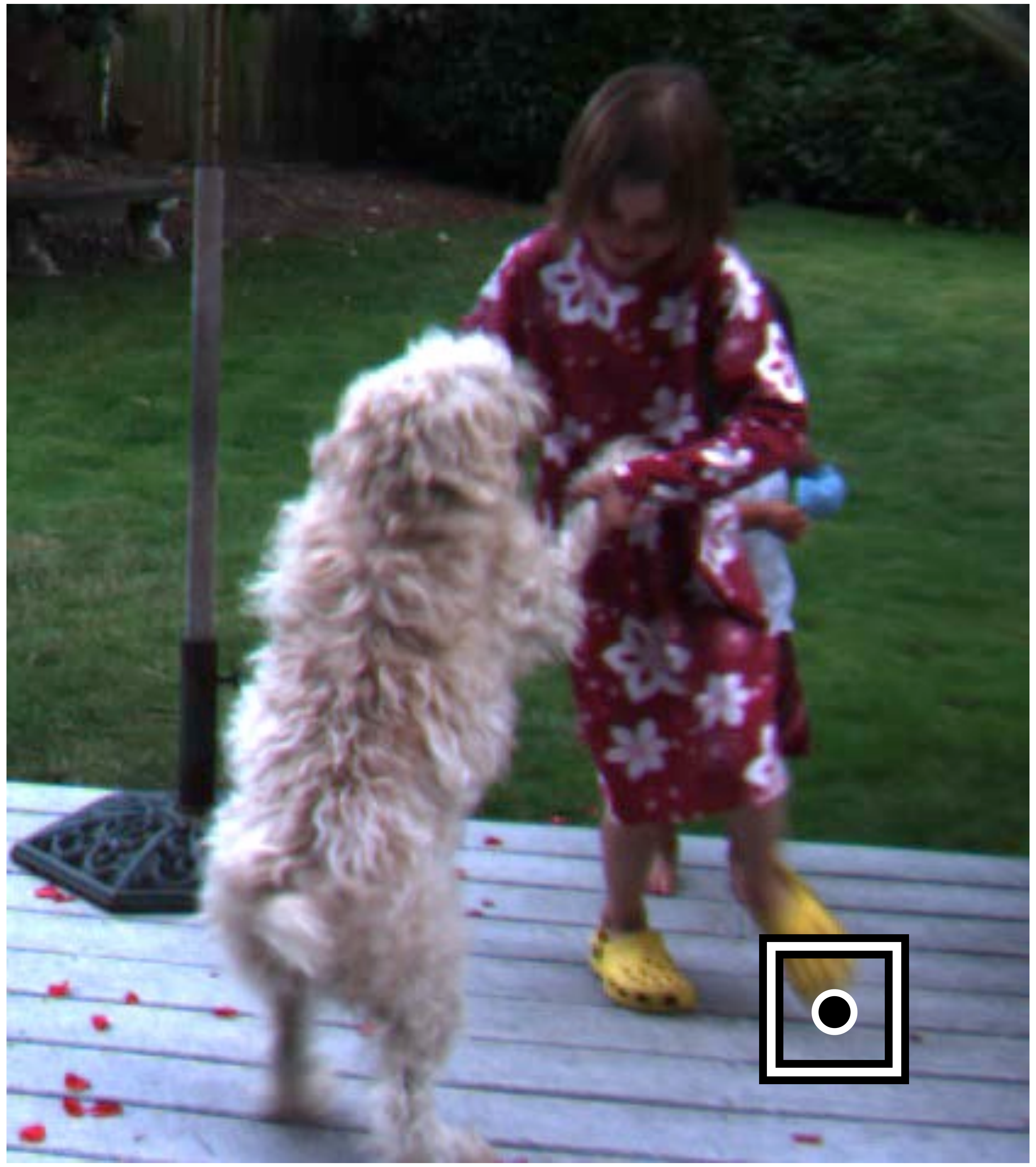
A point in image 1 gives a **line** in image 2

2-view **Rigid** Matching

1D search, points constrained to lie along epipolar lines

2-view **Non-Rigid** Matching

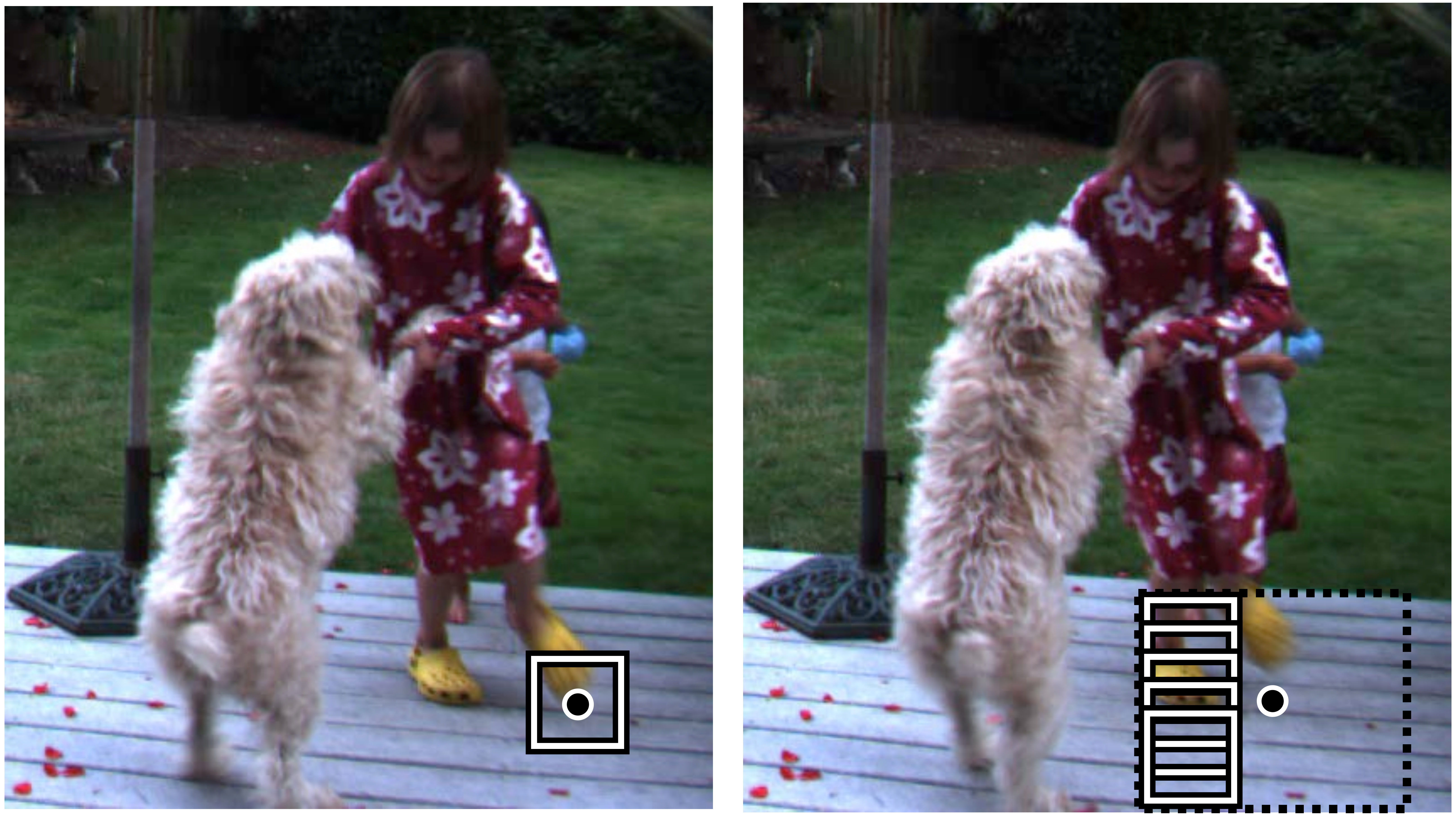
2D search, points can move anywhere in the image



[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

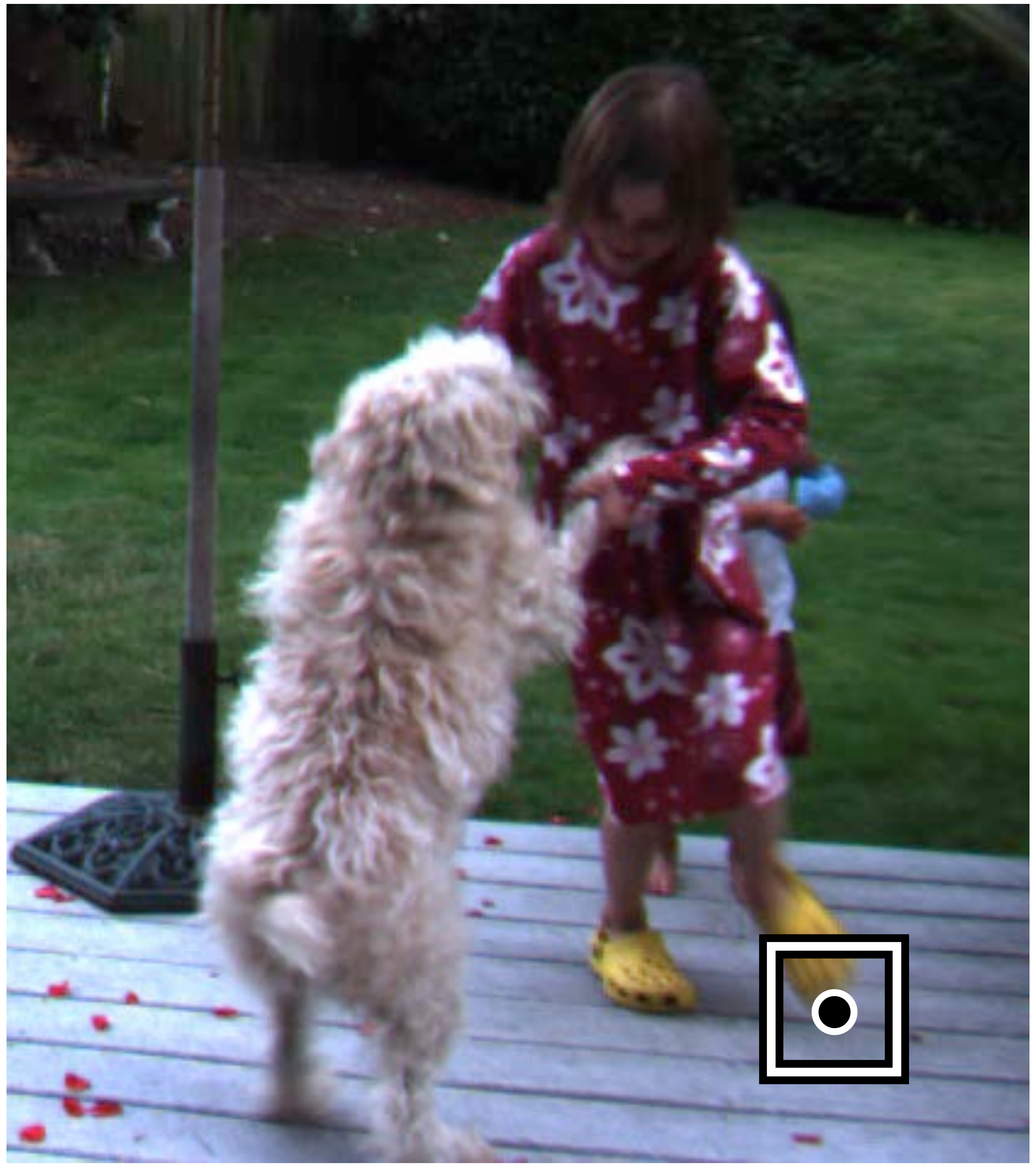
2D search, points can move anywhere in the image

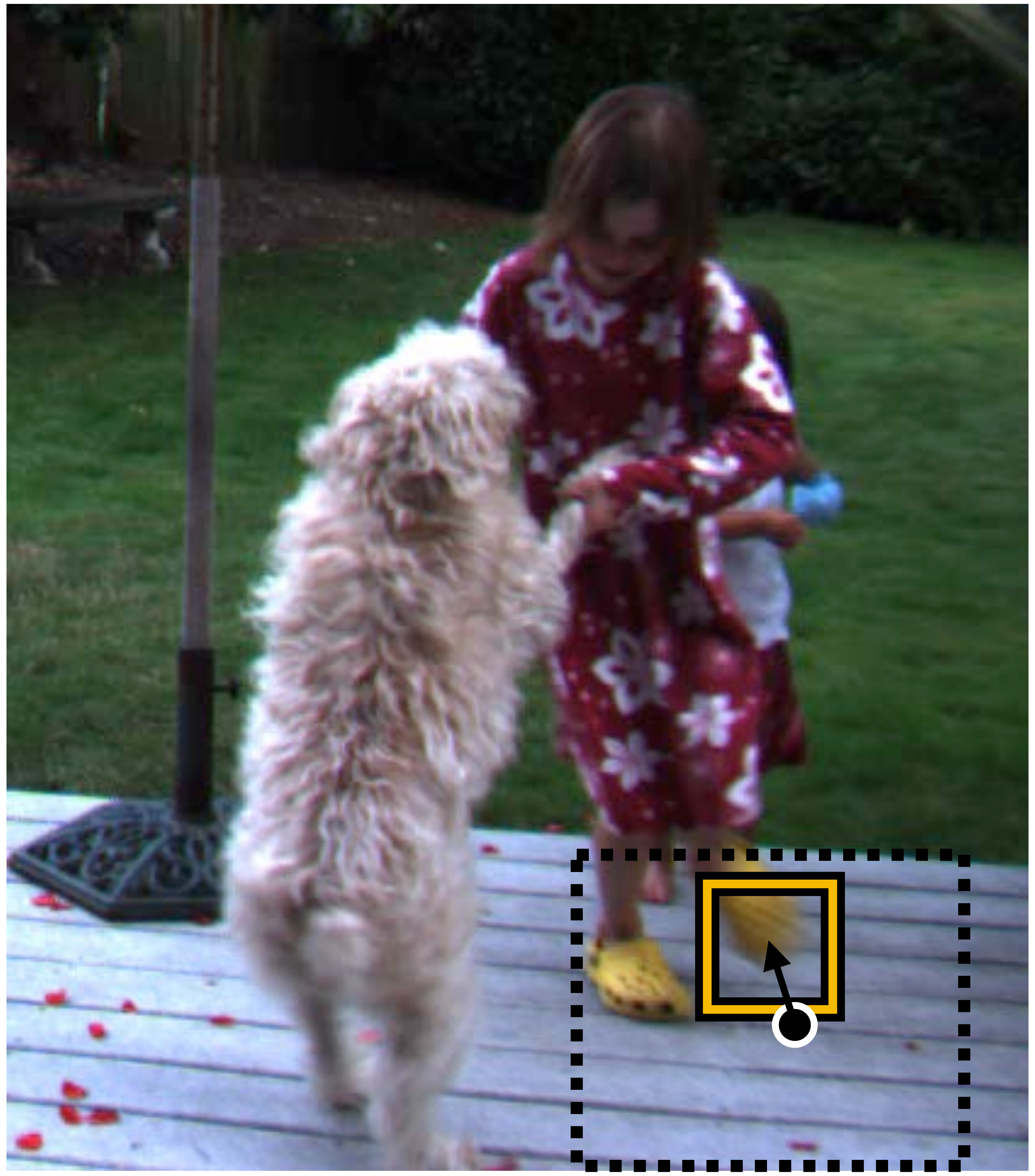


[vision.middlebury.edu/flow]

2-view **Non-Rigid** Matching

2D search, points can move anywhere in the image





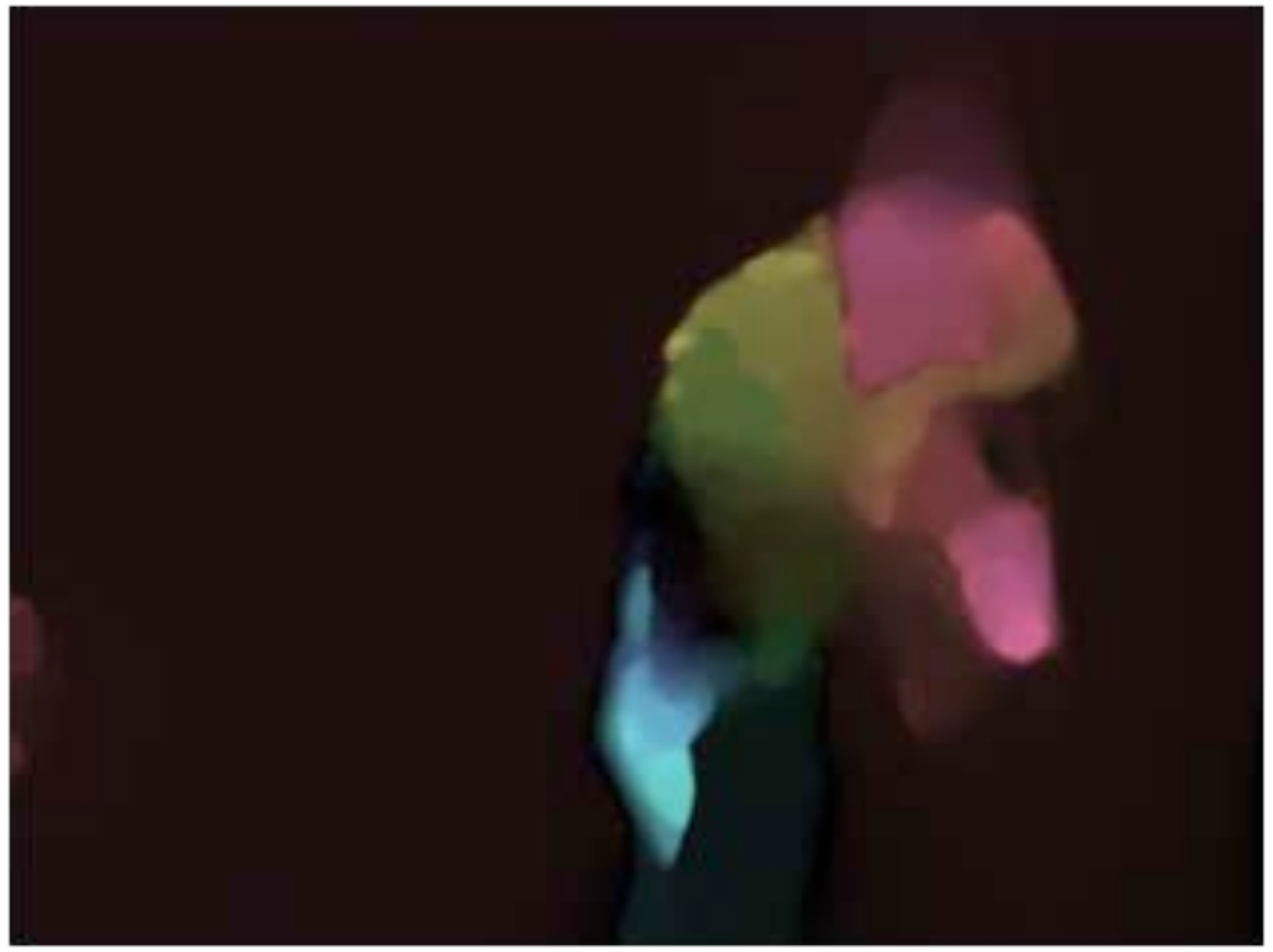
[vision.middlebury.edu/flow]

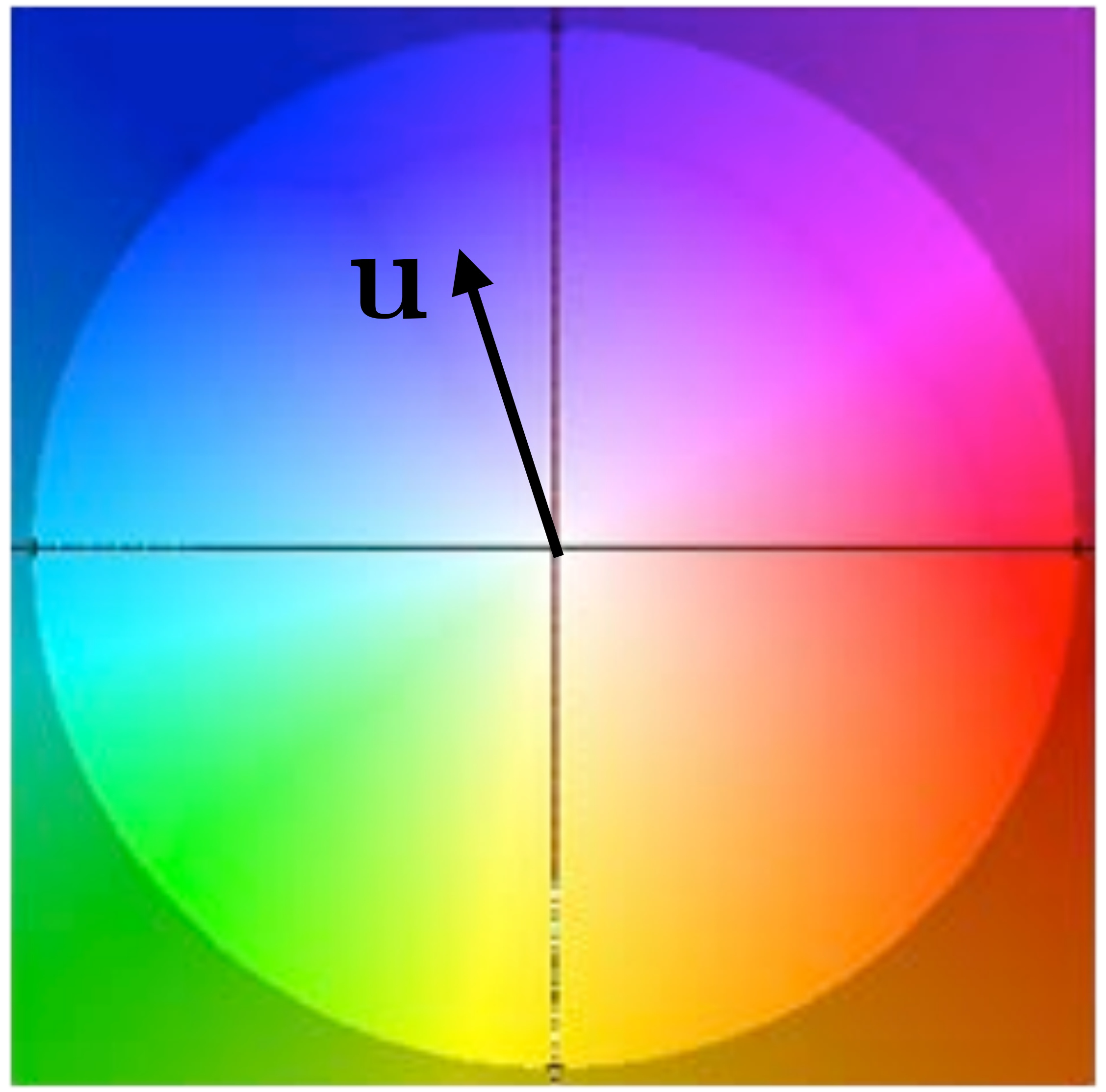
2-view **Non-Rigid** Matching

2D search, points can move anywhere in the image

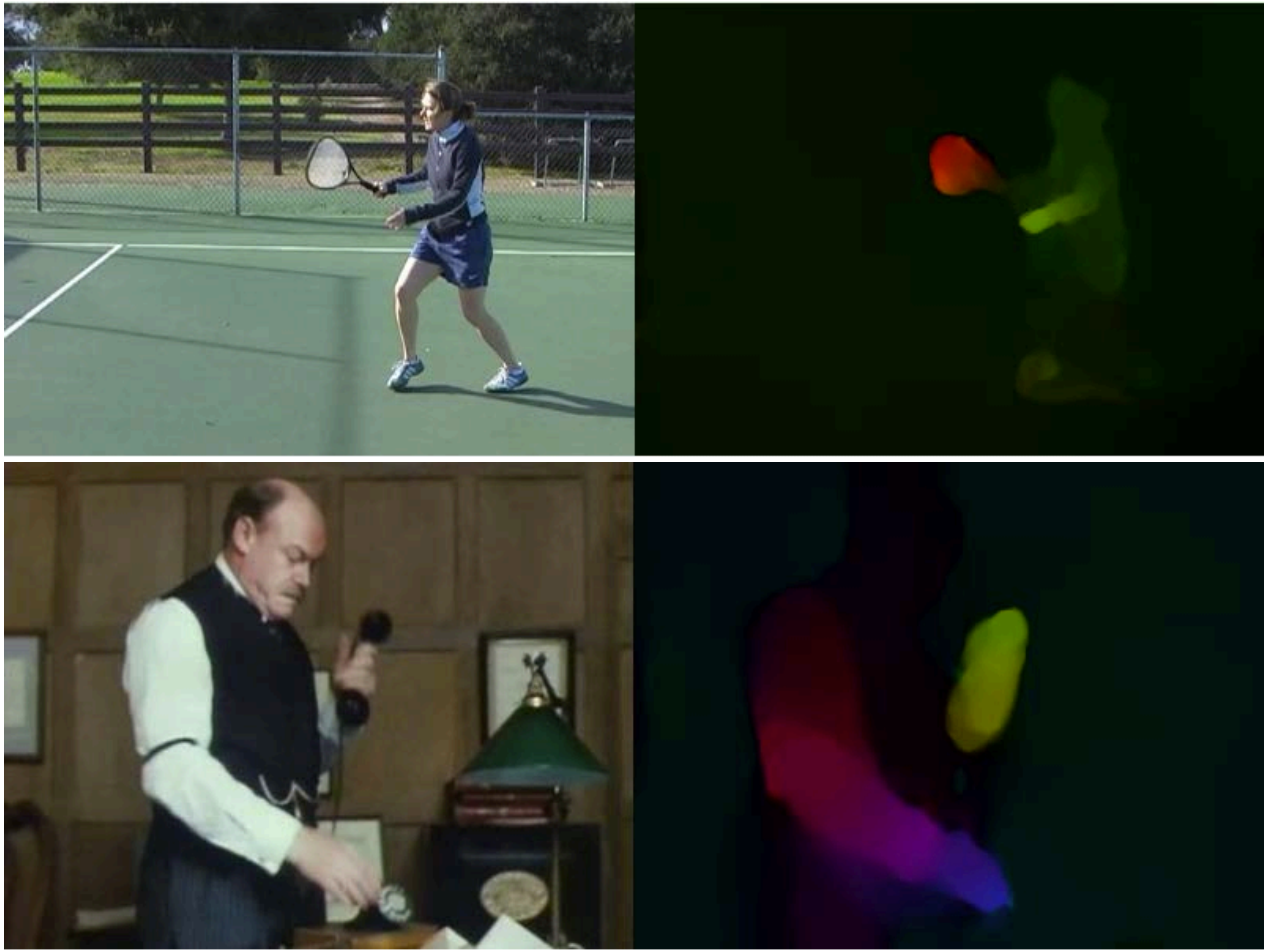
[vision.middlebury.edu/flow]

Optical Flow: Example 1





Optical Flow: Example 2



[Brox Malik 2011]

Optical Flow

Optical flow is the apparent motion of brightness patterns in the image

Problem:

Determine how objects (and/or the camera itself) move in the 3D world.
Formulate motion analysis as finding (dense) point correspondences over time.

Applications

— image and video stabilization in digital cameras, camcorders

— motion-compensated video compression schemes such as MPEG

— image registration for medical imaging, remote sensing

Dense vs Sparse Matching

Sparse: correspondence / depth estimated at discrete feature points, e.g., SIFT feature matches

Dense: correspondence / depth estimated at all locations, e.g., using stereo matching algorithms

Dense vs Sparse Matching

Optical Flow

In this lecture we'll focus on

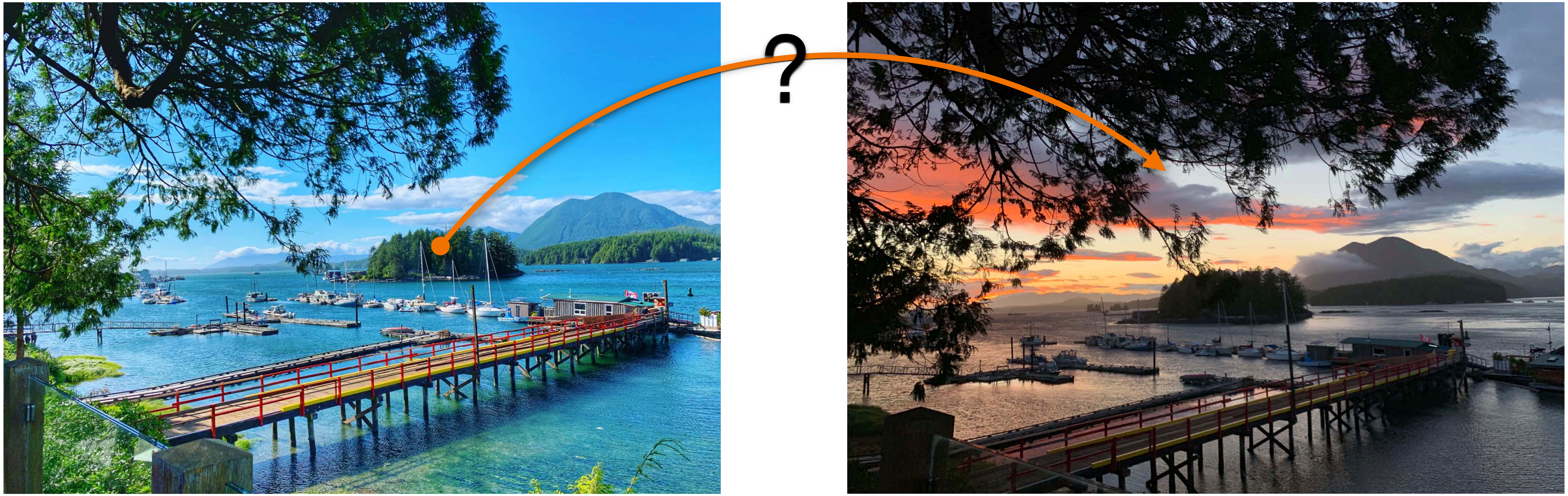
- **Dense flow** — compute correspondence / flow at every pixel
- **Short baselines** — assume small distances between frames, e.g., successive frames in a video

Wide baseline non-rigid matching algorithms do exist, but techniques are different (e.g., feature tracking)

[Z. Teed, Z. Deng, RAFT 2020]

Dense vs Sparse Matching in 2019

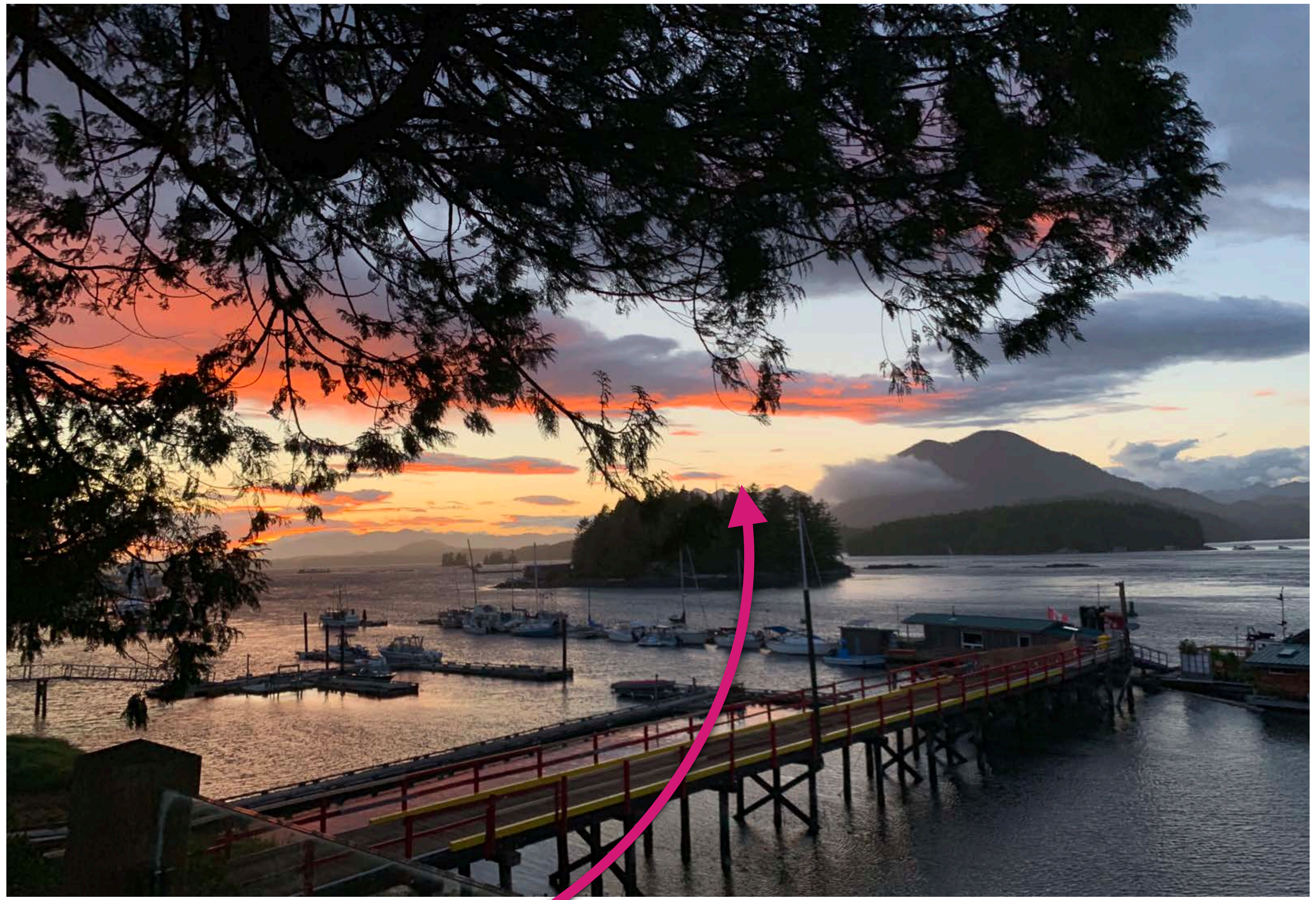
COTR: Correspondence Transformers



" where does the point go in the other image? "

Dense vs Sparse Matching in 2019

COTR: Correspondence Transformers



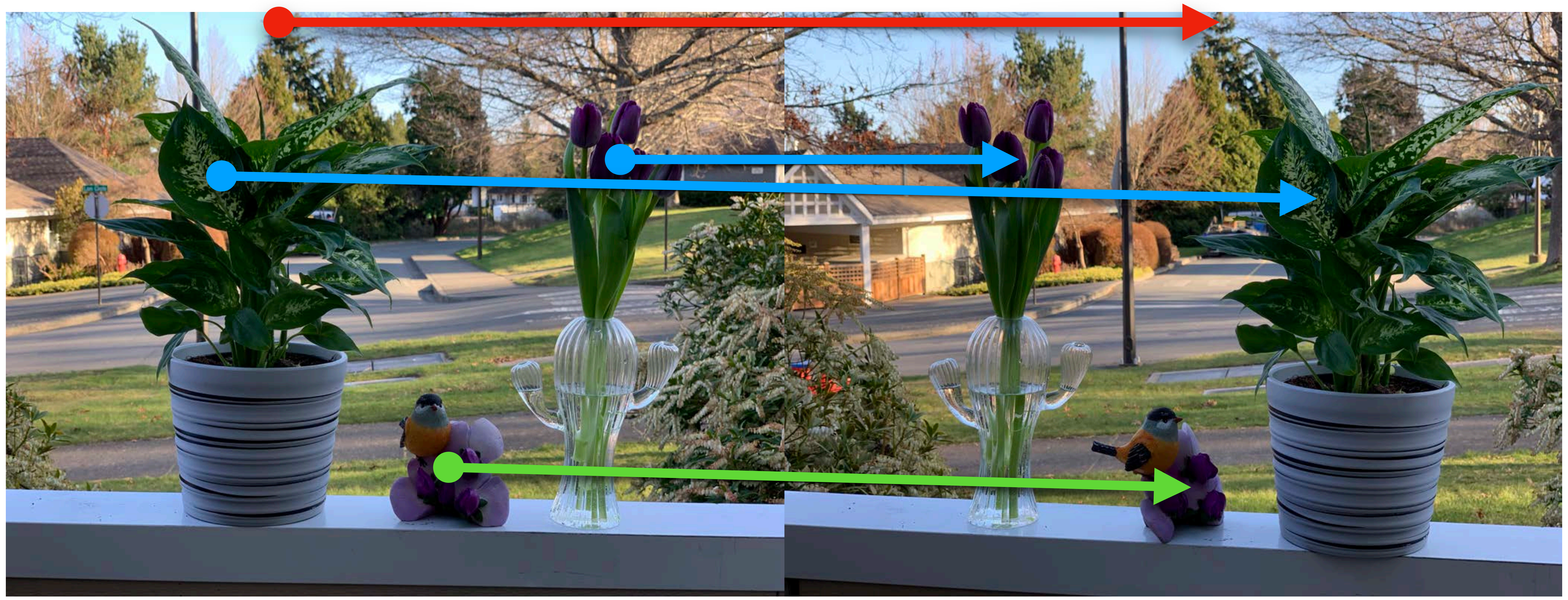
$$\mathbf{COTR}(x \mid I, I') = x'$$

Given an image pair and a query coordinate, it directly provides the corresponding coordinate in the other image.

Dense vs Sparse Matching in 2019

Solving both *sparse* and *dense* correspondences

Sparse



Solving **sparse** motions:

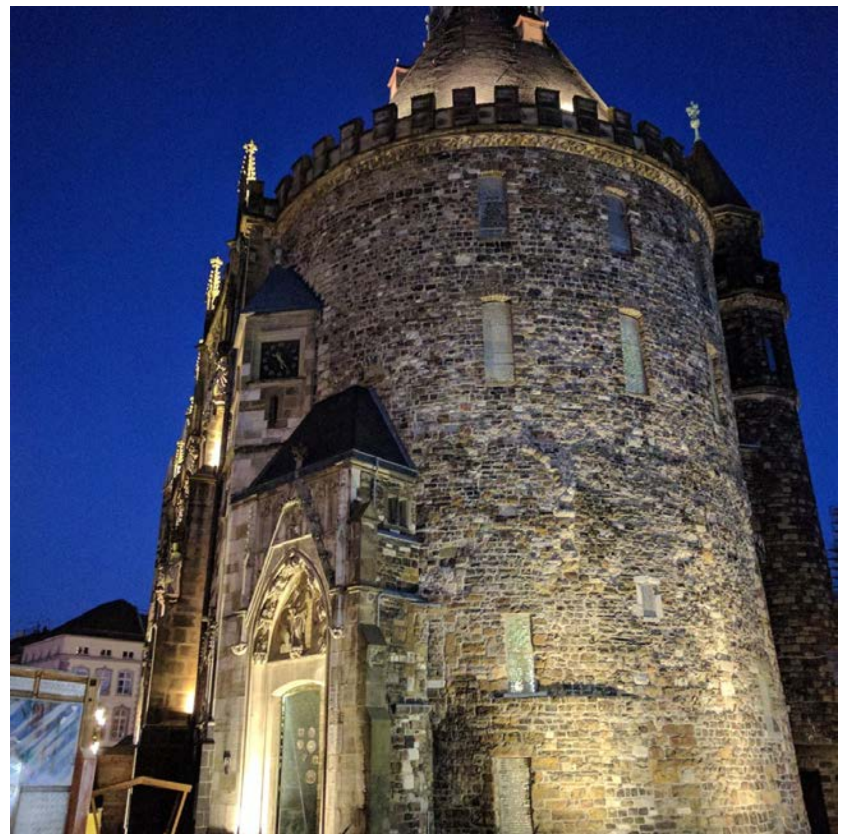
(actual results from our algorithm)

Red: Camera motion

Blue: Multi-object motion

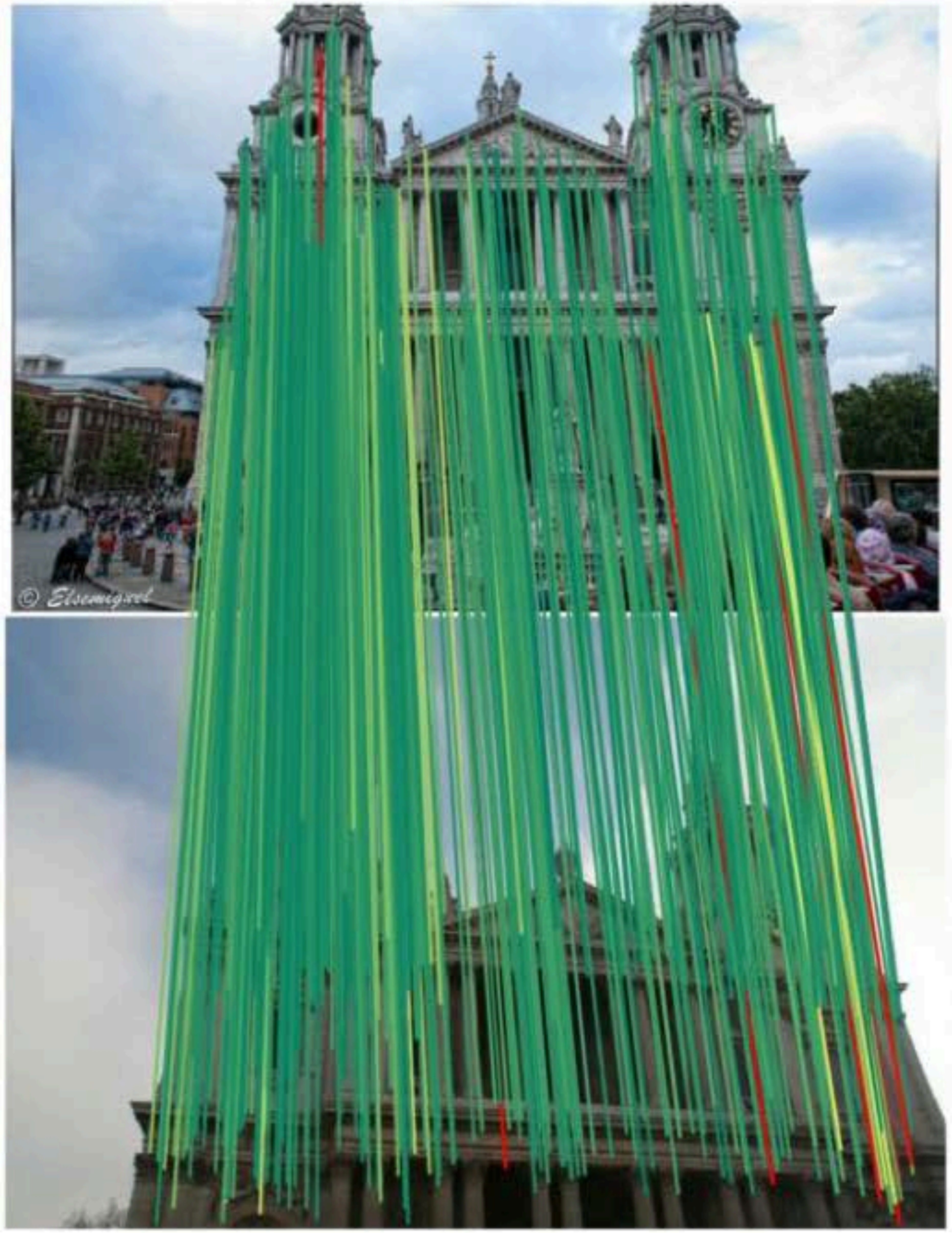
Green: Object-pose change

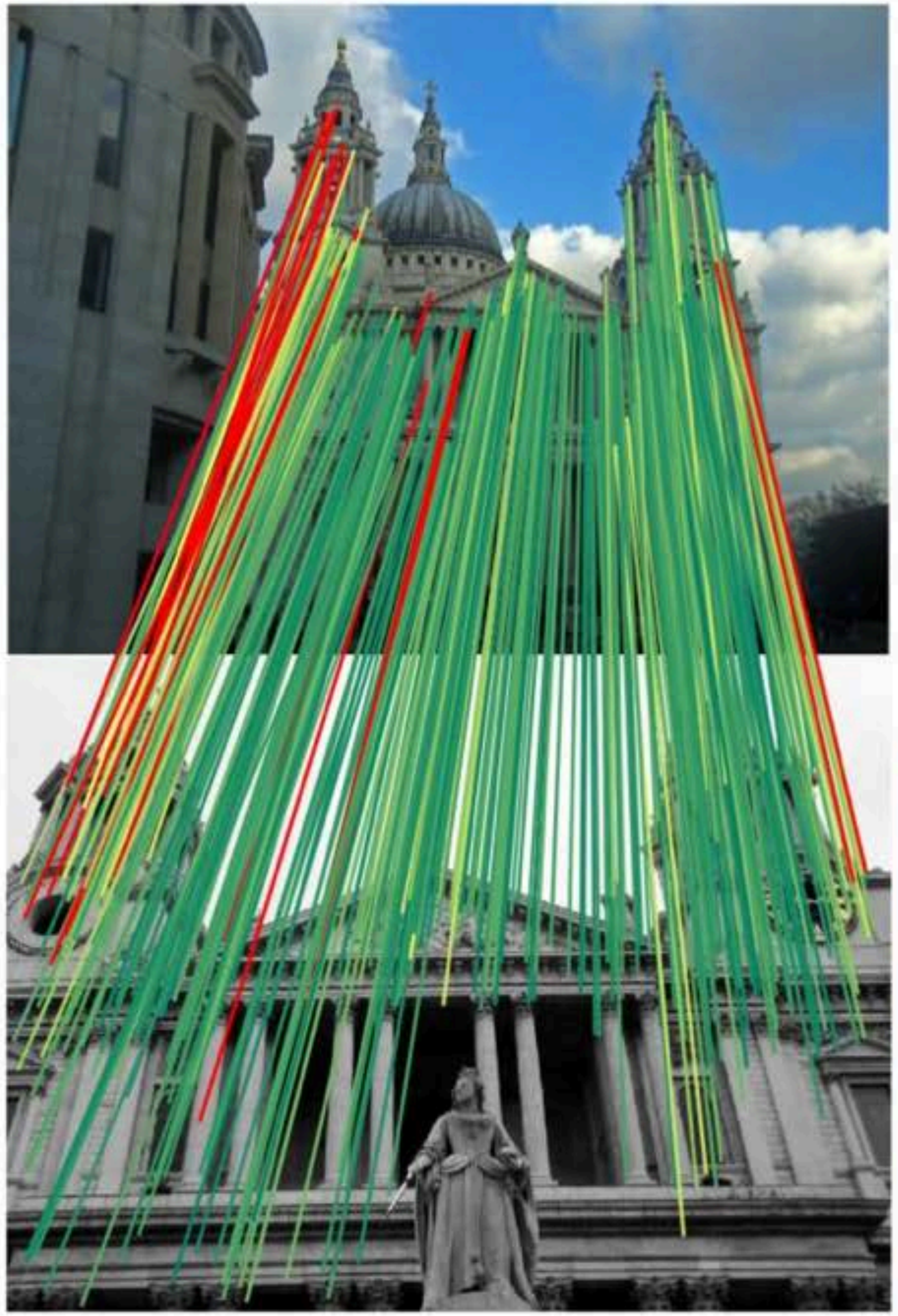
Dense

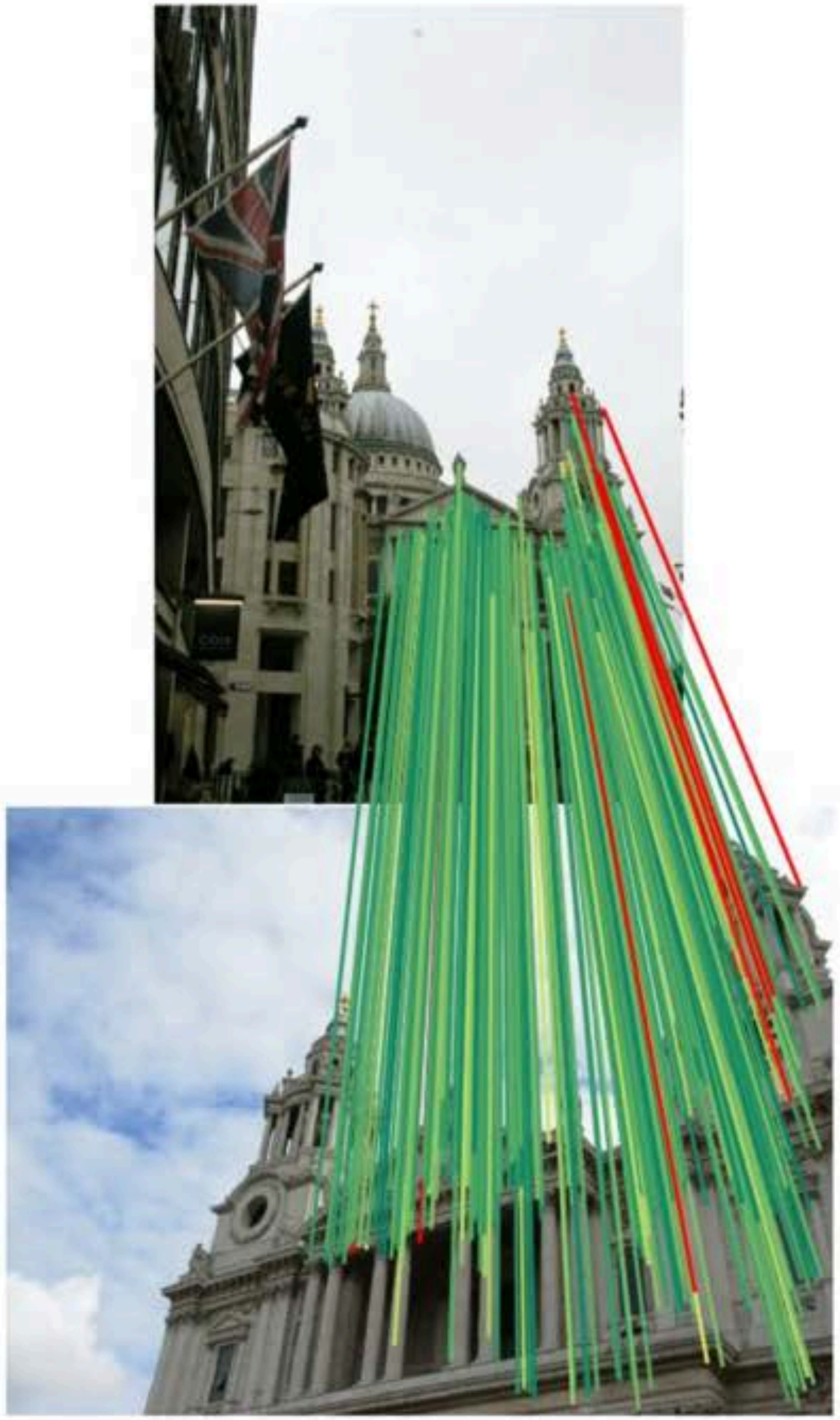
COTR(meshgrid |  , ) = 

Solving **dense** correspondence map and warping.

Dense vs Sparse Matching in 2019







Dense vs Sparse Matching in 2019

Images from [Teed and Deng, 2020], reproduced for educational purposes

Dense vs Sparse Matching

Optical Flow

In this lecture we'll focus on

- **Dense flow** — compute correspondence / flow at every pixel
- **Short baselines** — assume small distances between frames, e.g., successive frames in a video

Wide baseline non-rigid matching algorithms do exist, but techniques are different (e.g., feature tracking)

[Z. Teed, Z. Deng, RAFT 2020]

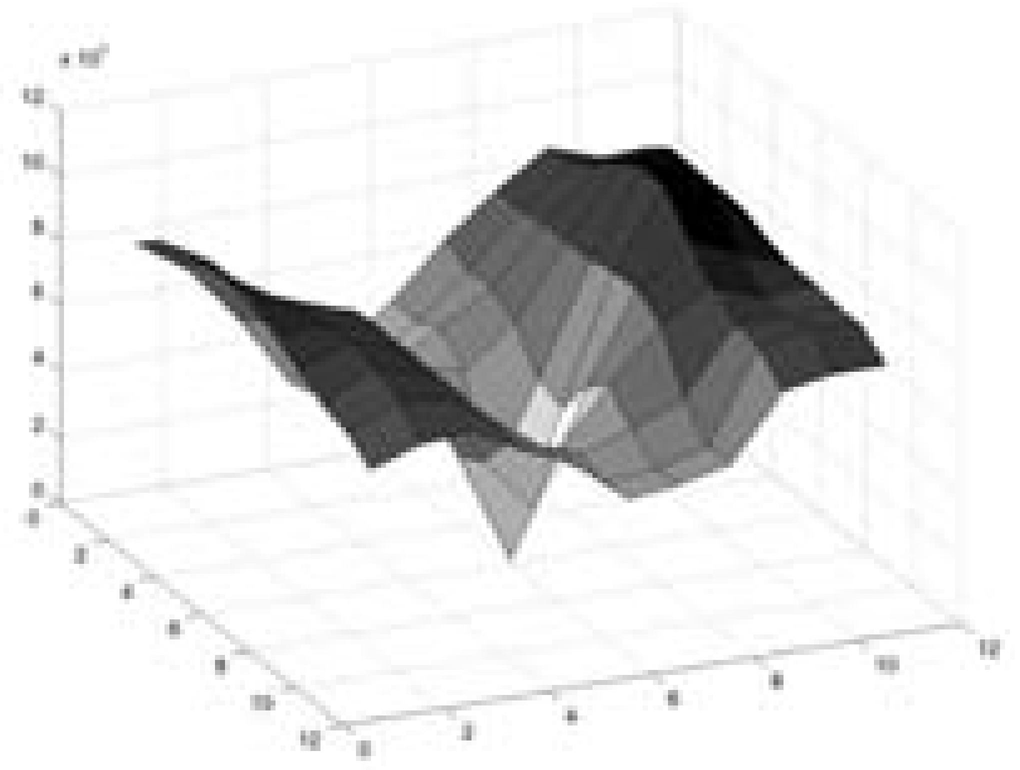
Lucas Kanade method

The previous algorithm suggested a discrete search over displacements/flow vectors **u**

We can do better by looking at the structure of the error surface:

$I_0(\mathbf{x})$

$I_1(\mathbf{x})$



$$e = |\mathbf{I}_1(\mathbf{x}+\mathbf{u}) - \mathbf{I}_0(\mathbf{x})|^2$$

15.1

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

How do we **compute** …

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

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

Forward difference
Sobel filter
Scharr filter
…

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

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

temporal derivative

Forward difference
Sobel filter
Scharr filter
…

Frame differencing

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Frame Differencing: Example

$$t+1 \qquad\qquad t \qquad\qquad I_t = \frac{\partial I}{\partial t}$$

$t+1$:

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

-

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

=

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

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

(example of a forward temporal difference)

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

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

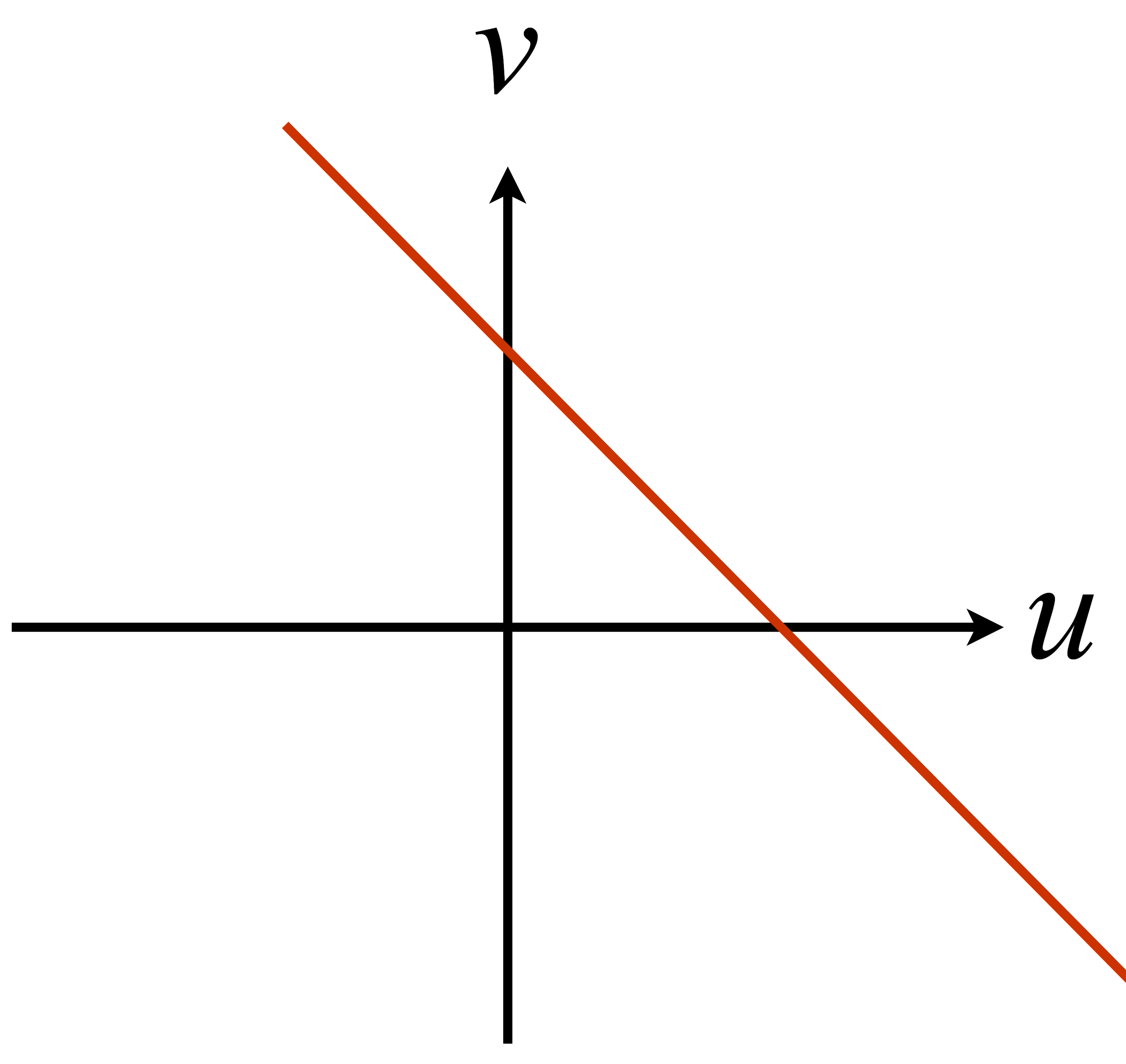
Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Optical Flow **Constraint Equation**

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

We have one equation in the two unknown components of velocity u, v

Many possible solutions for u, v — need more constraints or prior knowledge to solve



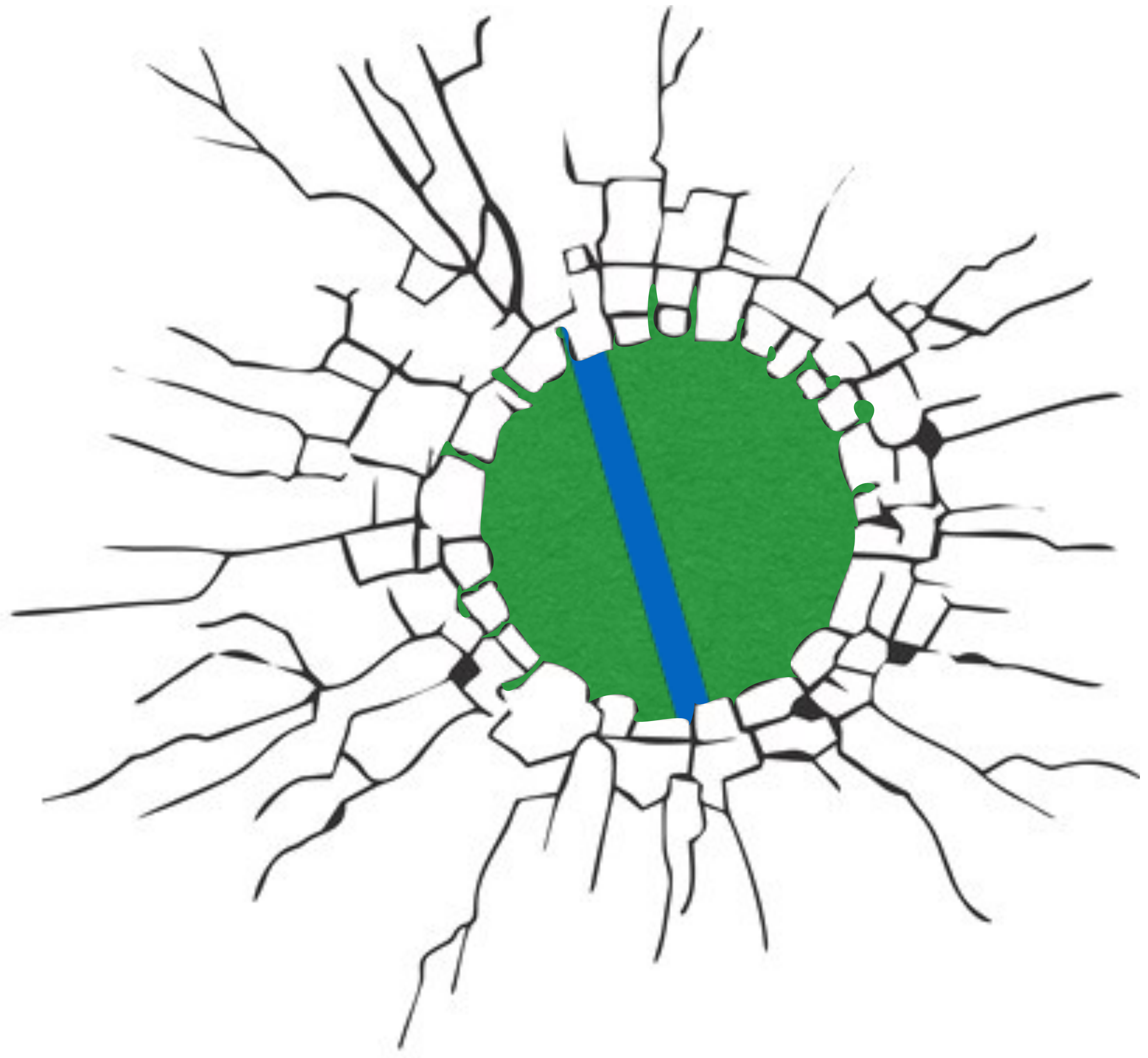
Equation determines a **straight line** in velocity space

Slide Credit: Ioannis (Yannis) Gkioulekas (CMU)

Flow **Ambiguity**

- The stripes can be interpreted as moving vertically, horizontally (rotation), or somewhere in between!
- The component of velocity parallel to the edge is unknown

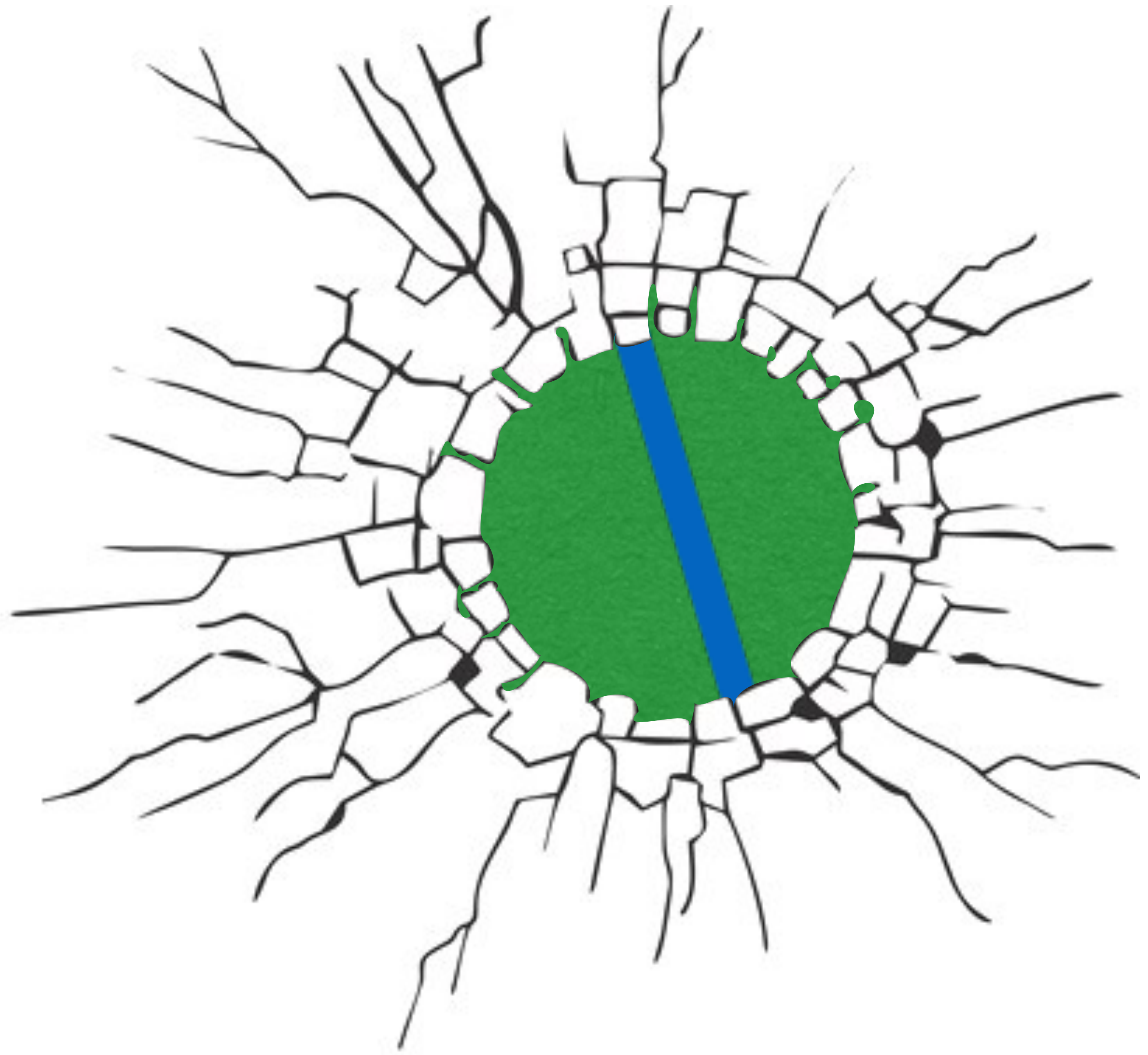
Aperture Problem



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

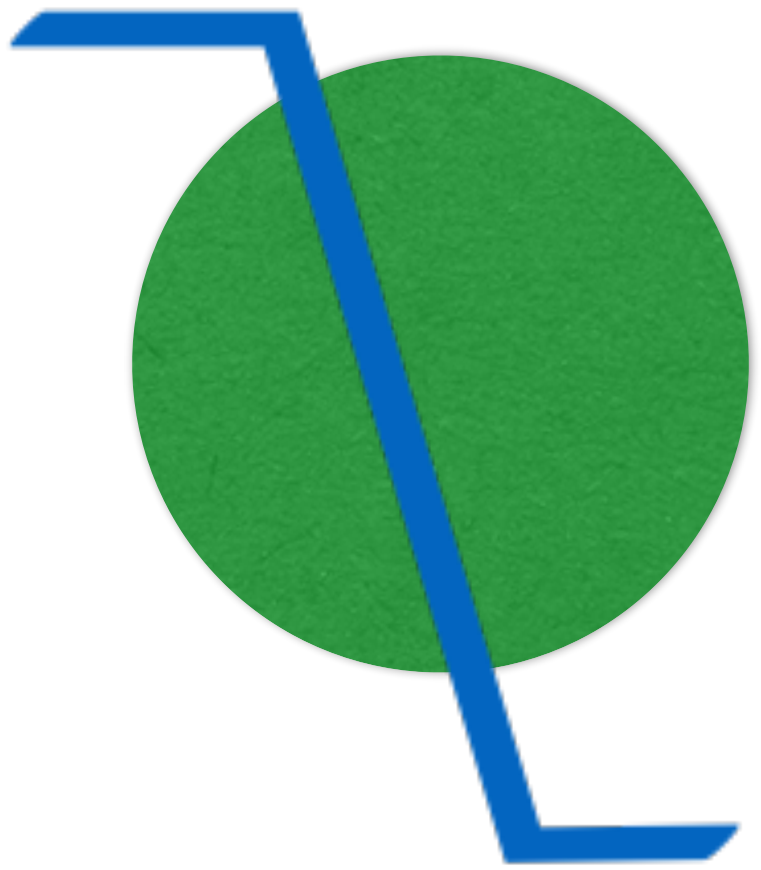


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

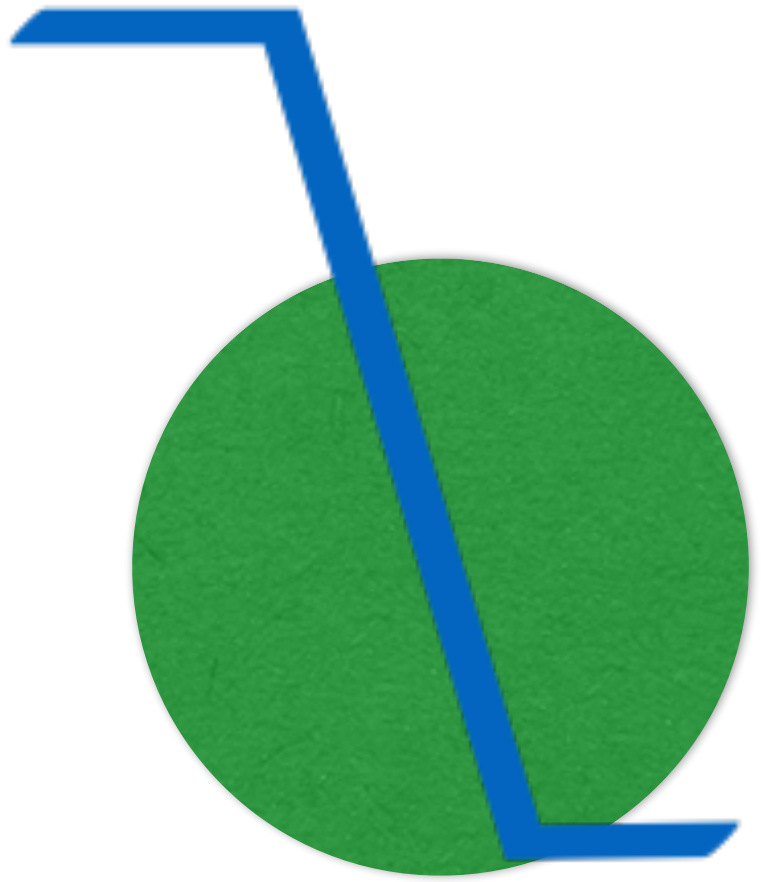


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

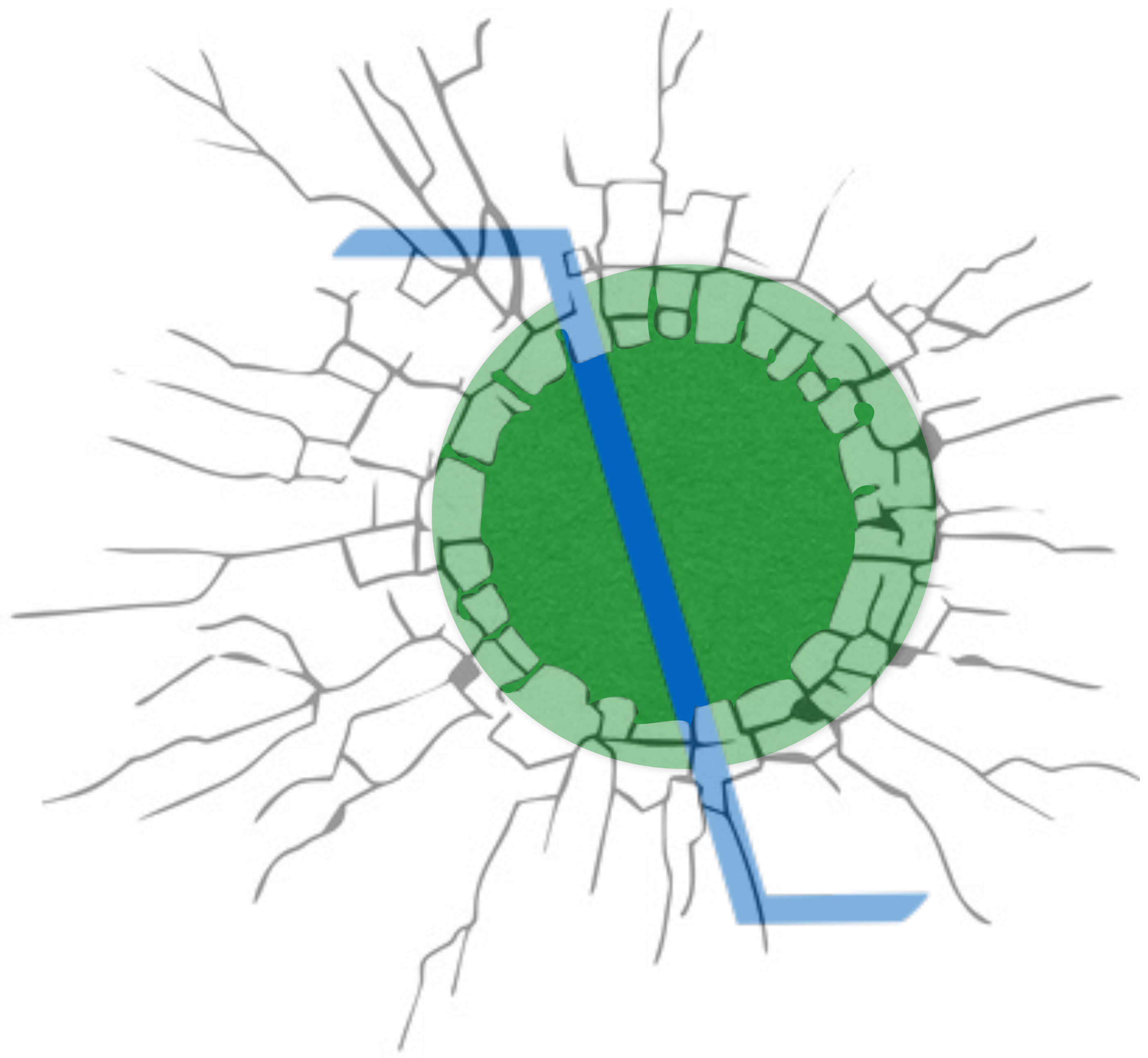


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Aperture Problem

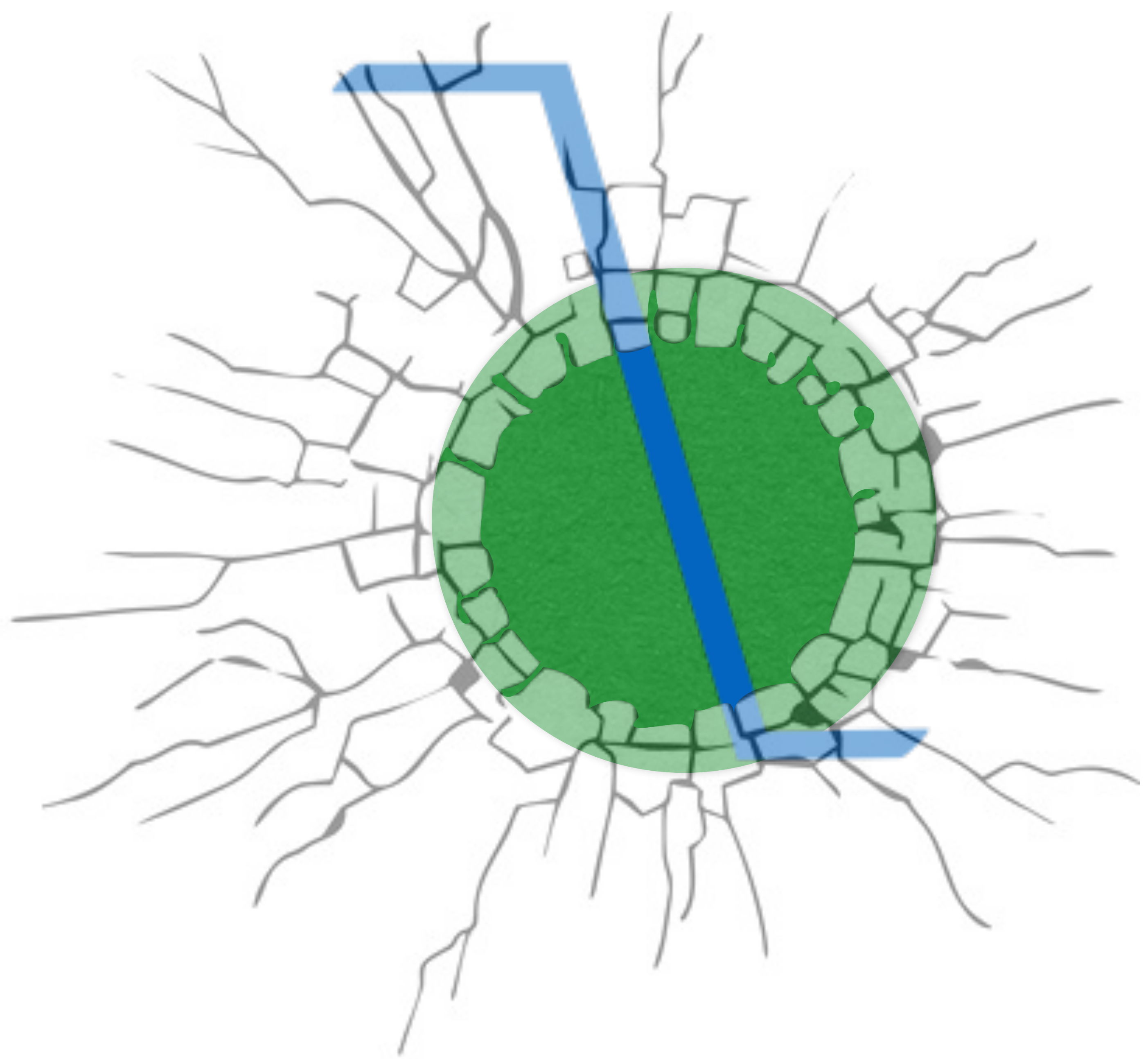
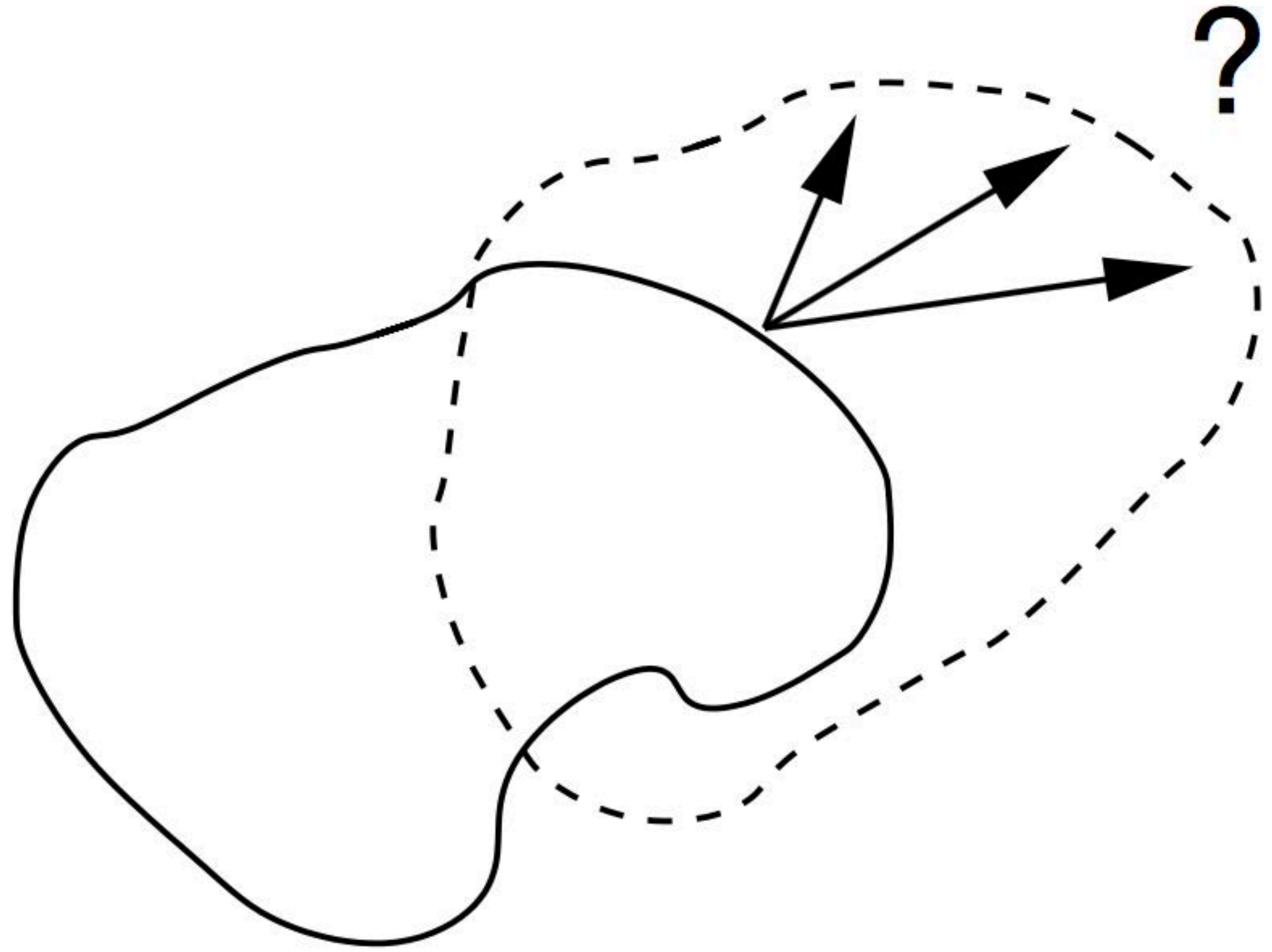


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

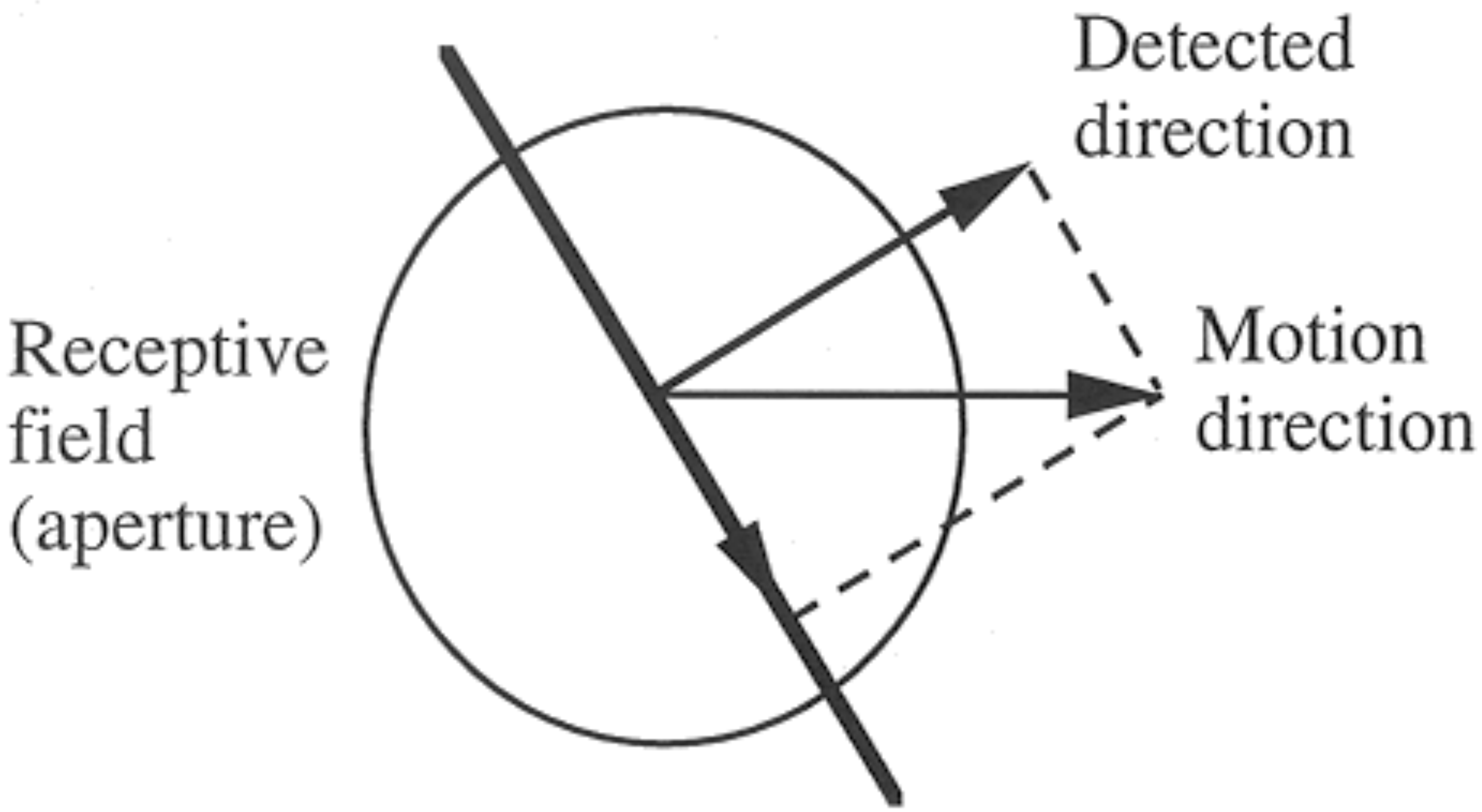
Aperture Problem



— Without distinct features to track, the true visual motion is ambiguous

— Locally, one can compute only the component of the visual motion in the direction perpendicular to the contour

Aperture Problem



— Without distinct features to track, the true visual motion is ambiguous

— Locally, one can compute only the component of the visual motion in the direction perpendicular to the contour

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

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

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

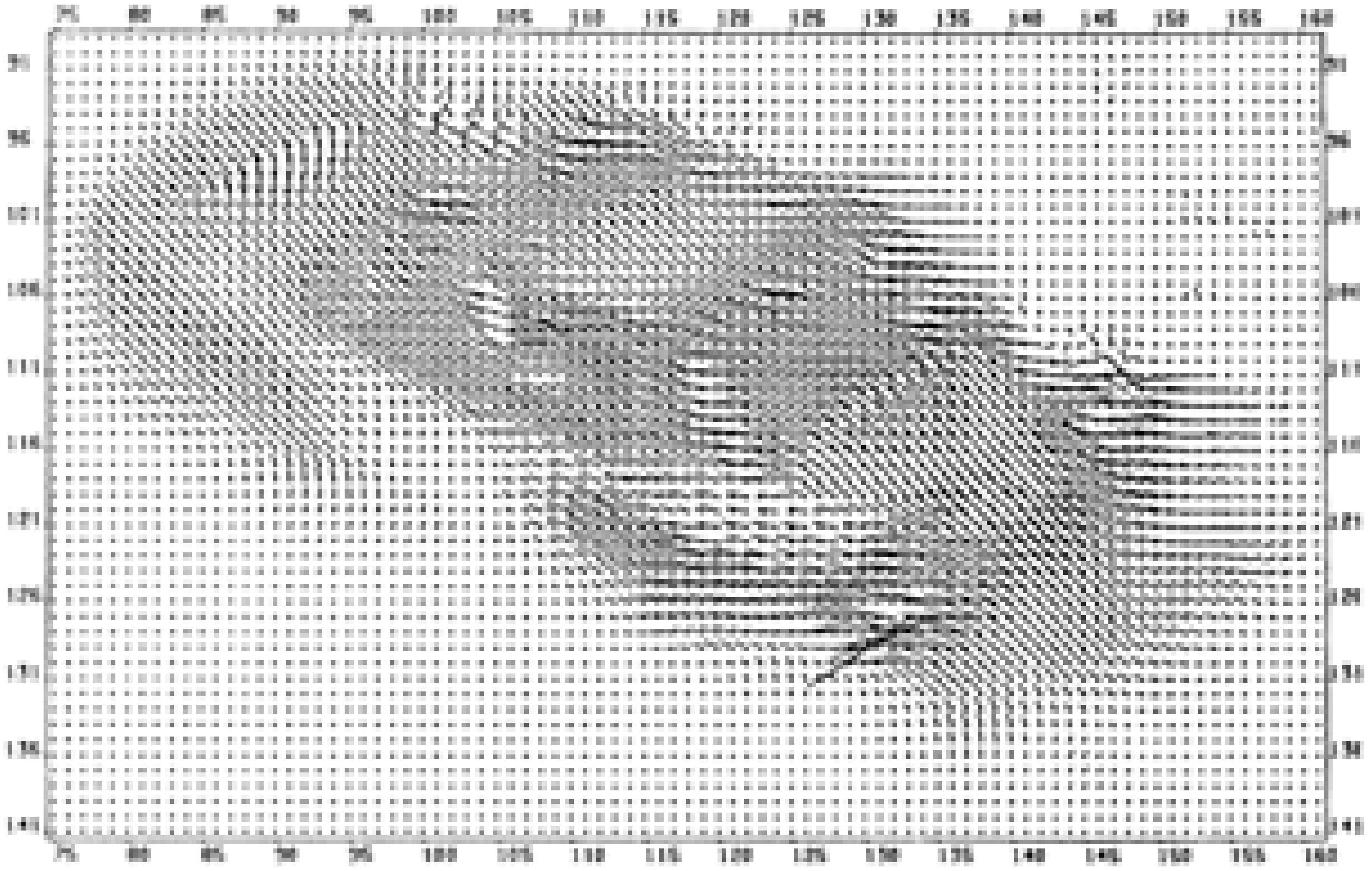
4. Windows are chosen s.t. that the rank of $\mathbf{A}^T\mathbf{A}$ is 2

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.

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

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]

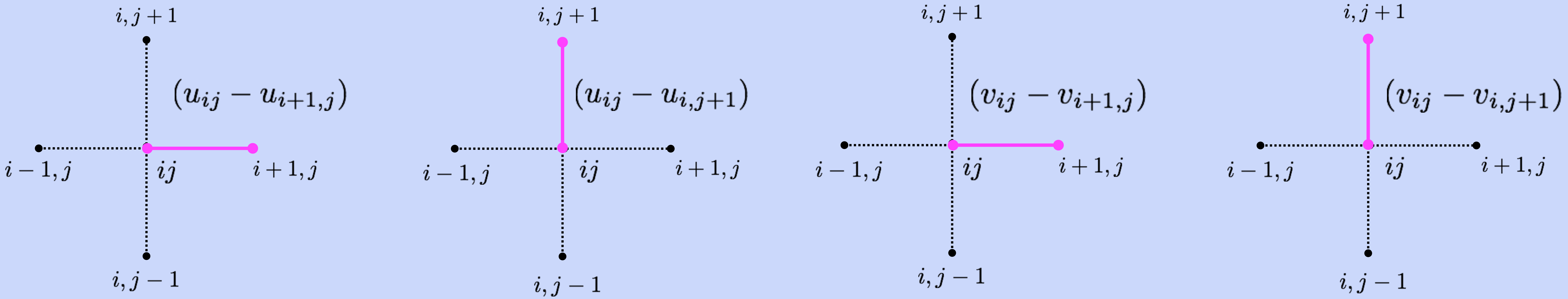
Horn-Schunck Optical Flow

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

$$I_1(\mathbf{x} + \mathbf{u}) \approx I_0(\mathbf{x})$$

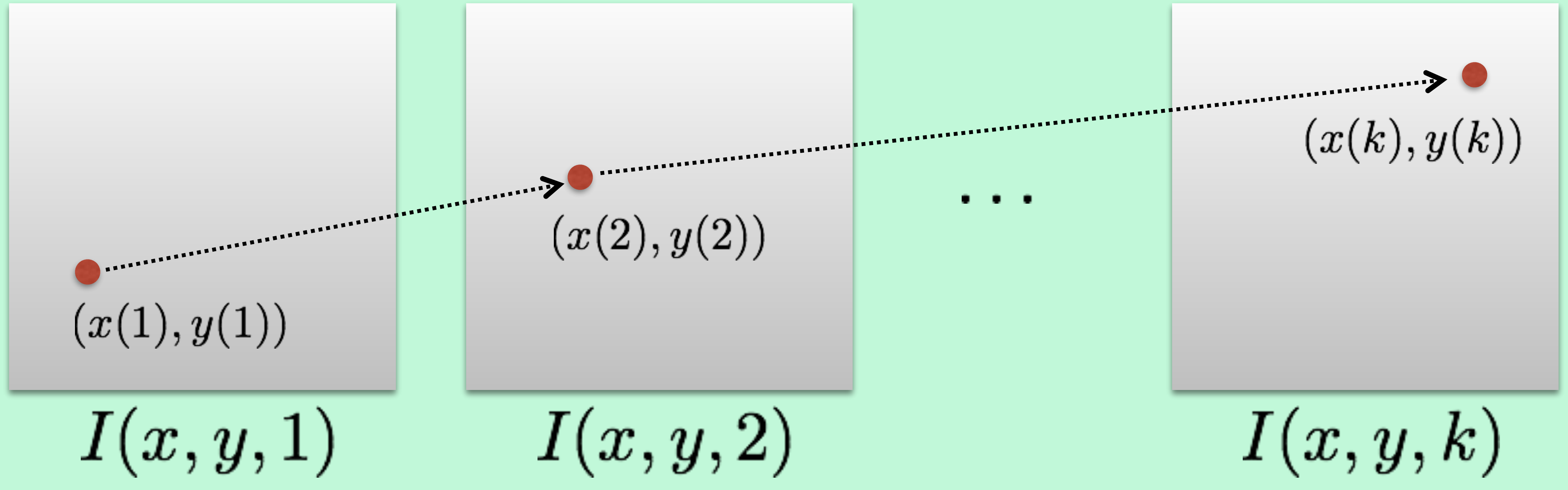
- This is called the **brightness constancy** assumption
- Taylor expansion for small motion at a single pixel → optical flow constraint

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

- 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 $\frac{dI(x, y, t)}{dt} = 0$. 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