CIS 5800

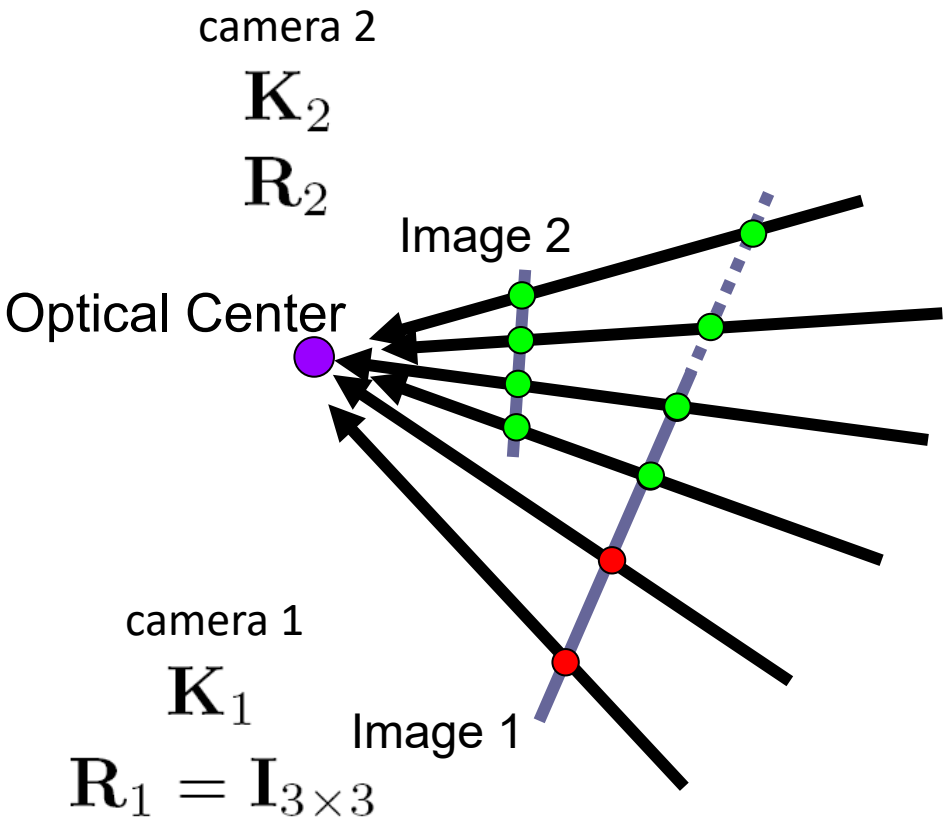
Machine Perception

Instructor: Lingjie Liu

Lec 16: April 2, 2025

Recap: Image stitching

For every pixel in image 2, we need to find the right pixel location to place it inside image 1.



3x3 homography

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \mathbf{K}_1 \mathbf{R}_1 \mathbf{R}_2^{\mathrm{T}} \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

image coords (in image 1)

image coords (in image 2)

3D ray direction (in cam 2 coordinates)

3D ray direction in world coordinates

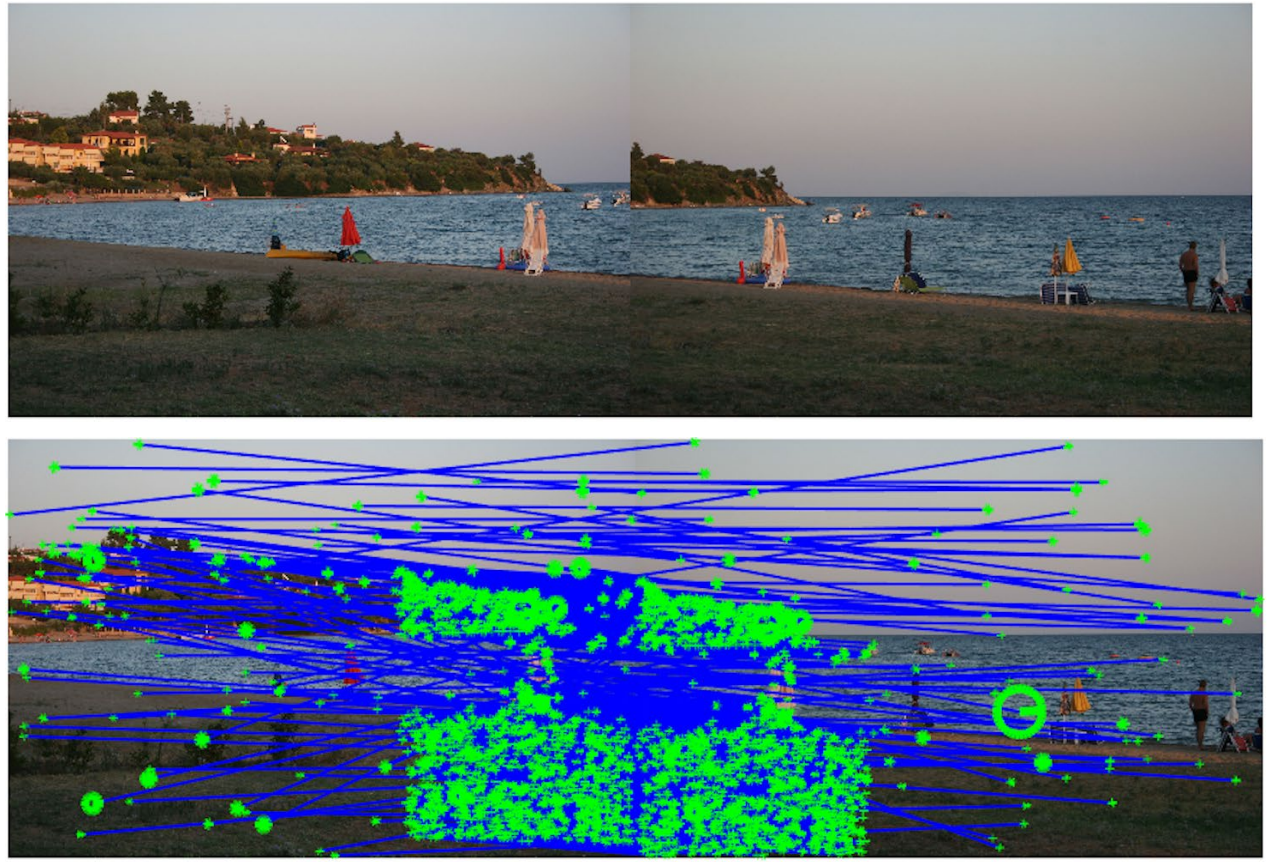
3D ray direction in camera 1 coordinates

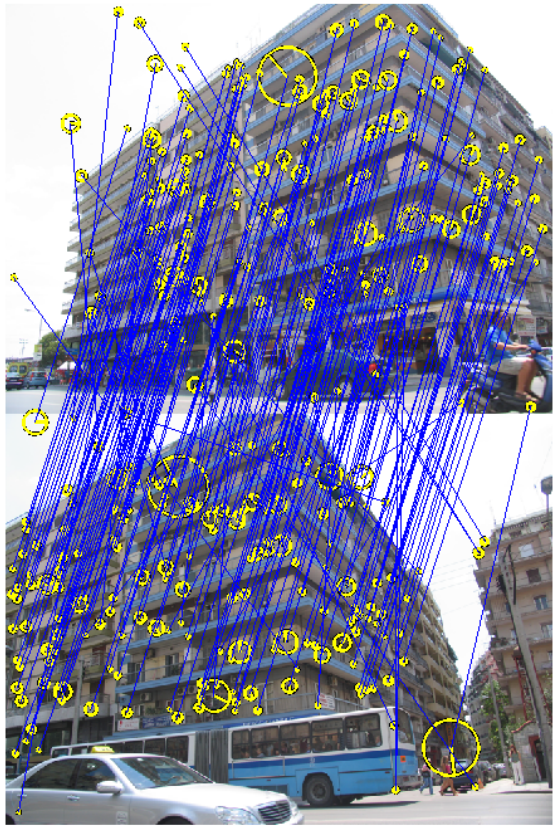
Project through camera 1!

Homography even though not restricting to imaging a plane!

Recap: Correspondences are not clean!

Or we might have to group them !
(two planes=>two homographies)

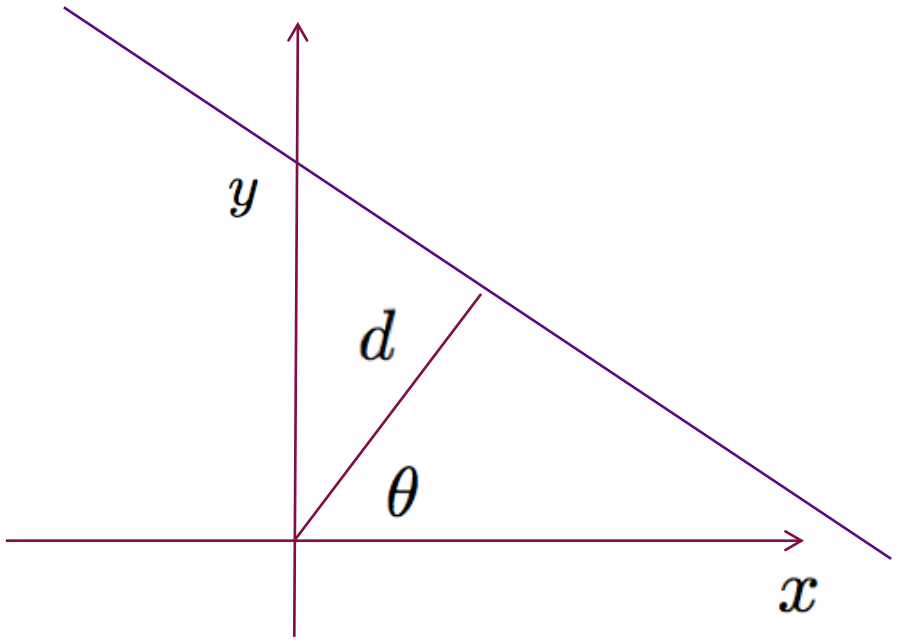




Recap: A Case Study: Fitting A Line To Data

Given data (x_i, y_i) belonging to a line

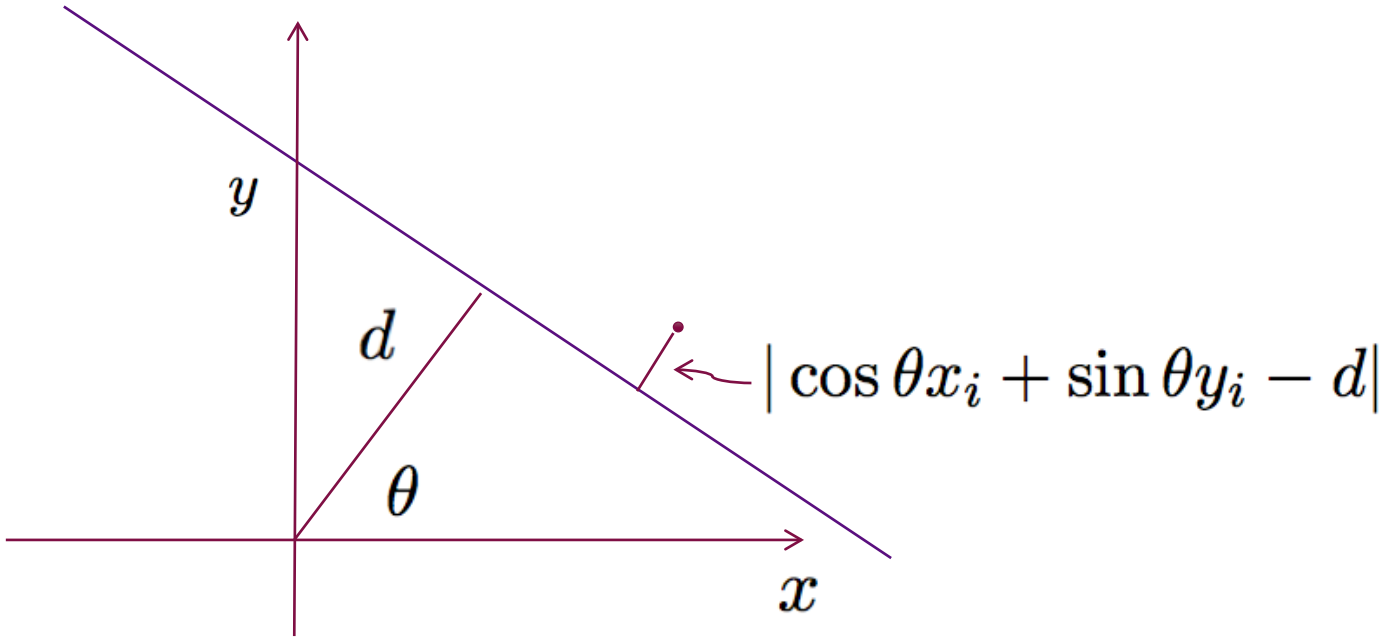
$$x\cos\theta + y\sin\theta = d$$



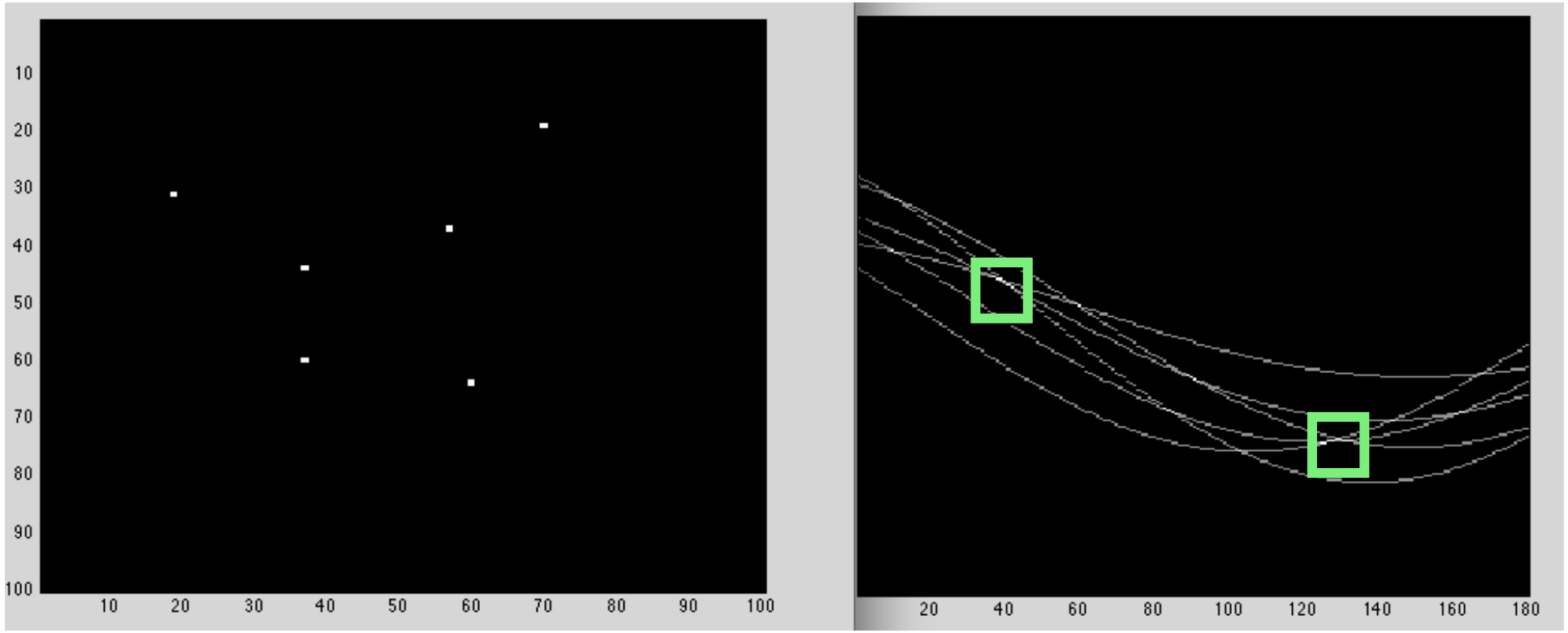
Recap: Finding the line by optimization

To solve for the line, we could minimize the mean squared error:

$$\underset{\theta \in [0,2\pi), d \geq 0}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i \cos\theta + y_i \sin\theta - d)^2$$



Recap: Hough Voting



Recap: Disadvantages of Hough Transforms

Hough transforms are great and simple-to-implement approach for grouping k samples into n “groups” (e.g. lines, circles, VPs etc.) + outliers that don’t belong to any group.

- Note: Needs a bounded parameter space to allow a finite discrete set of hypotheses, and careful discretization.
- Main drawback: Poor scaling. Becomes intractable when # parameters(unknowns) for the target groupings is more than 3 or at most 4 in practice.
  - So, e.g. to fit homographies (8 parameters), completely intractable.

Alternative solution that scales much better: RANSAC!

RANSAC

Random Sample Consensus for detecting “inliers and outliers”

Goal: Detecting inliers and outliers

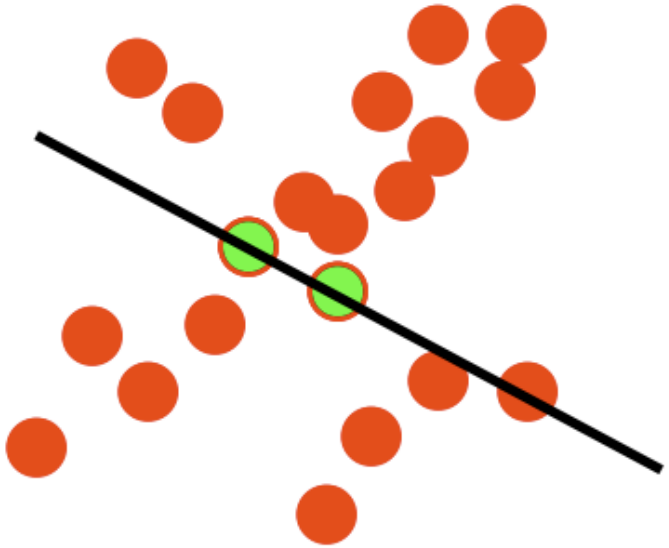
- Given that we are detecting ***one group*** (e.g. one line, or one circle, or one homography) in some noisy data that includes outliers, how to do this?
  - Easier than the multiple group-finding problem that Hough tries to tackle.
  - But hopefully, our solution will be more efficient in high-dimensional settings, where ”groups” will have many parameters.

Back to basics: a single line

Given that we are fitting a *single* line, say, can we figure out inliers and outliers?

First, One Possible Hough Voting Approach

- Set up a vote accumulator with D entries, one corresponding to each discrete line hypothesis under consideration.
- For each pair out of C_n^2 pairs of points:
  - Find the corresponding line l using a minimal set of points with M points
  - Vote 1 for the closest discrete line hypothesis $l'_{disc} \approx \delta$ from the hypothesis set, 0 for all else. Add all votes to the accumulator.
- Going over all entries in the accumulator, pick the one with the highest votes.

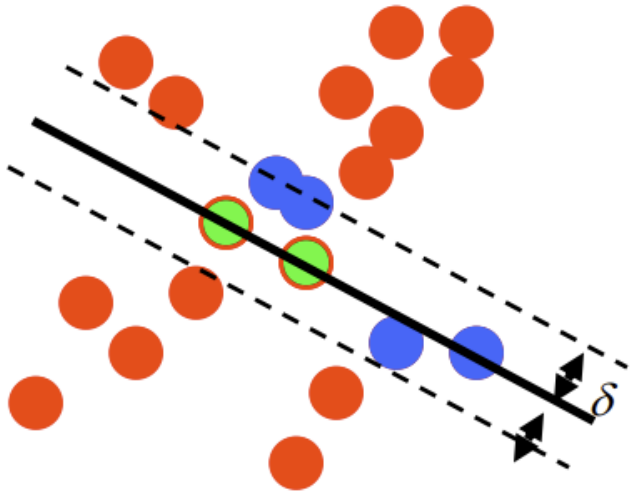


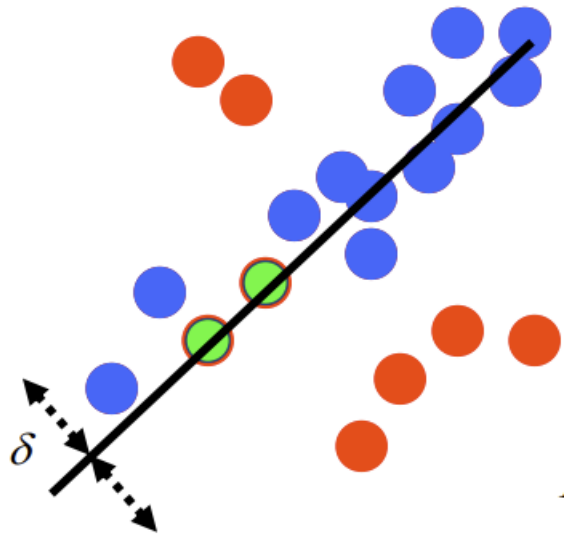
Minimal sample sets in geometry

- Plane: 3 points
- Circle: 3 points
- Ellipsoid: 5 points
- Line: 2 points
- Homography: 4 2D->3D point correspondences on a 3D plane
- P3P: 3 2D->3D point correspondences in 3D

First, One Possible Hough Voting Approach

- Set up a vote accumulator with D entries, one corresponding to each discrete line hypothesis under consideration.
- For each pair out of C_n^2 pairs of points:
  - Find the corresponding line l using a minimal set of points with M points
  - Vote 1 for the closest discrete line hypothesis $l'_{disc} \approx \delta$ from the hypothesis set, 0 for all else. Add all votes to the accumulator.
- Going over all entries in the accumulator, pick the one with the highest votes.





Not-Quite RANSAC: Sample Consensus (Non-Random)

- Set maximum inlier count $Max = 0$
- For each pair out of C_n^2 pairs of points:
  - Find the corresponding line l using a minimal set of points with M points
  - Check how many other points approximately lie on this line ("inliers").
  - If $n_{inliers} > Max$, set $Max = n_{inliers}$ and set best candidate to l

This is still a problem though.

No need to discretize the hypothesis space into a finite set.
No need to store and track the full hypothesis set.

So, does not blow up in complexity as the hypothesis space grows (more unknown parameters, finer discretization …)

$n_{inliers} = 6$

$n_{inliers} = 14$

RANdom SAmple Consensus or RANSAC

Graphics and Image Processing

J. D. Foley Editor

Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Martin A. Fischler and Robert C. Bolles
SRI International

and analysis conditions. Implementation details and computational examples are also presented.

Key Words and Phrases: model fitting, scene analysis, camera calibration, image matching, location determination, automated cartography.

CR Categories: 3.60, 3.61, 3.71, 5.0, 8.1, 8.2

I. Introduction

We introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The application discussed, the location determination problem (LDP), is treated at a level beyond that of a mere example of the use of the RANSAC paradigm; new basic findings concerning the conditions under which the LDP can be solved are presented and a comprehensive approach to the solution of this problem that we anticipate will have near-term practical applications is described.

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed

Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography

MA Fischler, RC Bolles - Communications of the ACM, 1981 - dl.acm.org Paperpile

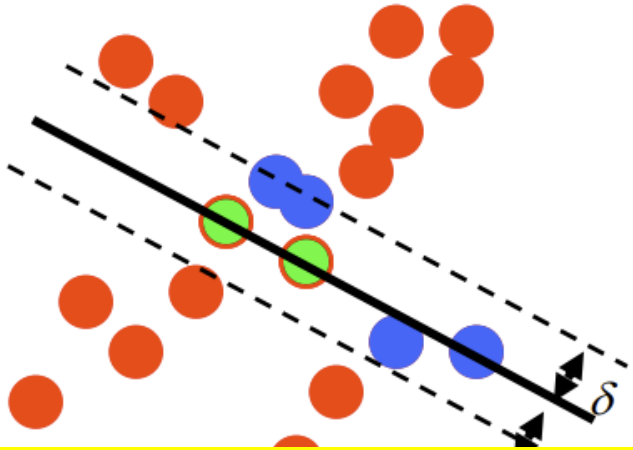
… We introduce a new paradigm, **Random Sample Consensus** (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The …

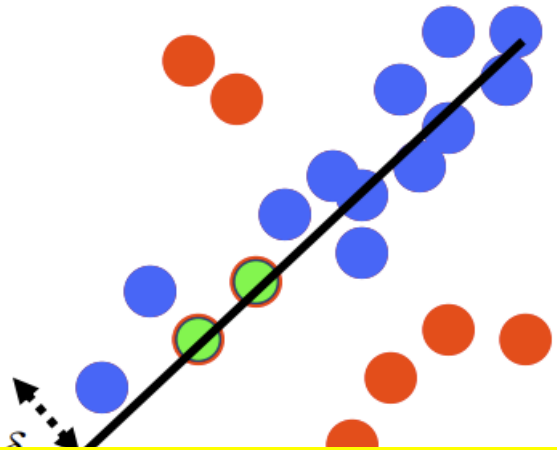
☆ Save 99 Cite Cited by 29840 Related articles All 12 versions Import into BibTeX

Fischler, 1981

RANSAC in the line fitting setting

- Set maximum inlier count $Max = 0$
- For some k randomly chosen pairs out of C_2^n pairs of points:
  - Find the corresponding line l using a minimal set of points with M points
  - Check how many other points approximately lie on this line ("inliers").
  - If $n_{inliers} > Max$, set $S = n_{inliers}$ and set best candidate to l





How to choose the number of trials k?

Probabilistic analysis of RANSAC

If this happens, we will find the correct answer!

- Suppose the probability of any given sample being a ***true inlier*** is ϵ
- Suppose the minimal set has M points (e.g. 2 for line fitting)
- Q: What is the probability that the minimal sample set is *all inliers*?

A: ϵ^M

- Q: In k iterations, what is the probability of *never* finding an *all-inlier* set?

A: $(1-\epsilon^M)^k$

If this happens, RANSAC would fail!

In other words, for given M, ϵ, if we try k times, the probability that we hit the correct answer is:

$$1-(1-\epsilon^M)^k$$

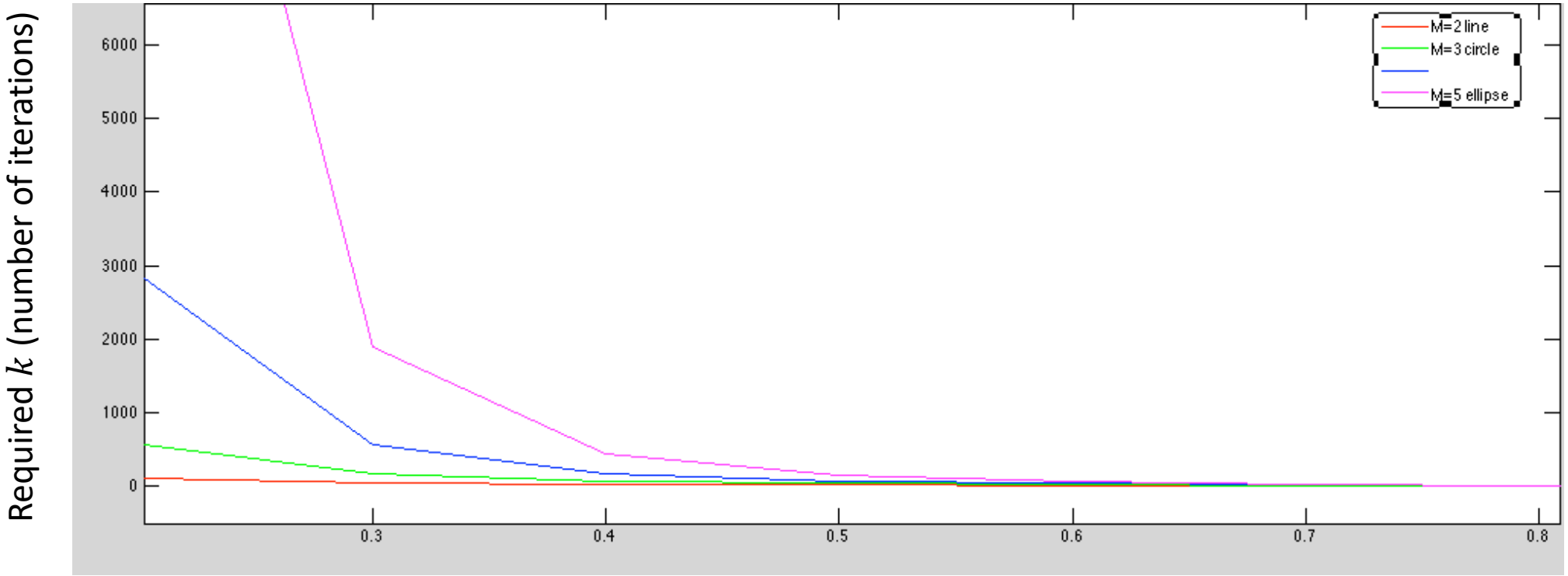
Probability of success for RANSAC!

How many times do we have to try?

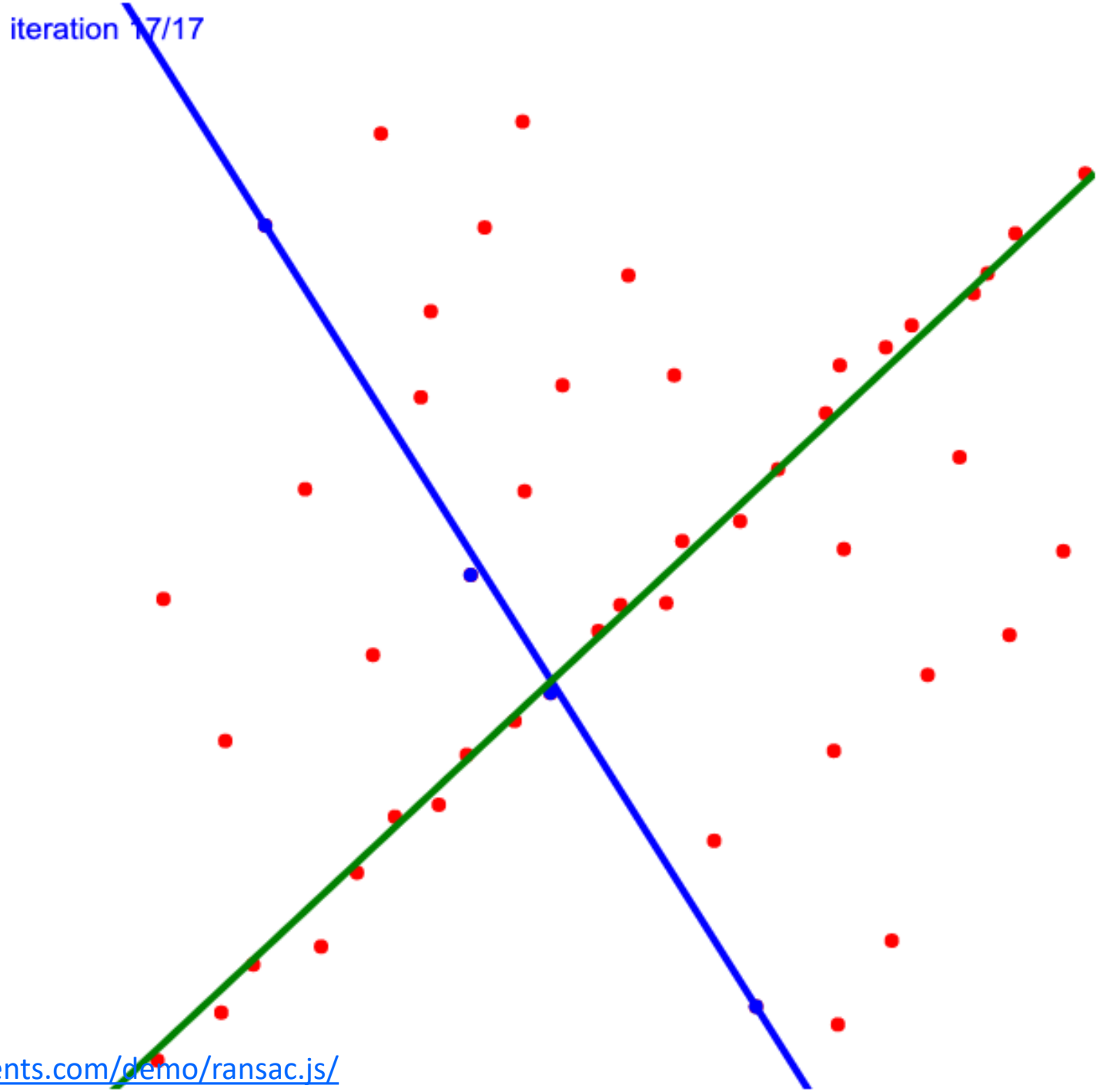
$1 - (1 - \epsilon^M)^k$

$$k = \frac{\log(1-p)}{\log(1-\epsilon^M)}$$

Number of iterations needed to find the correct solution with probability 0.9 for the case of M=2 (line), 3 (circle), 4 (rectangle), 5 (ellipse)

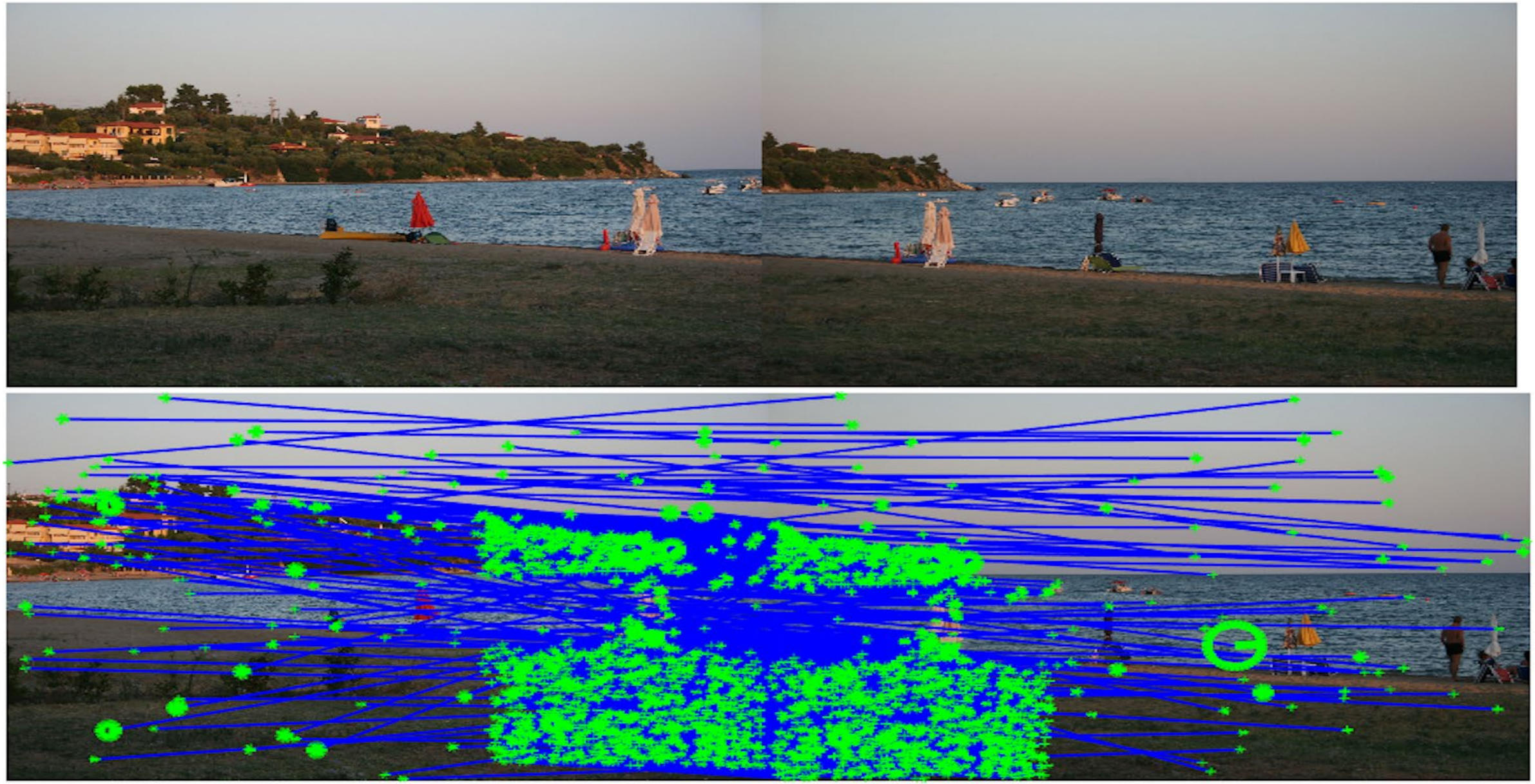


Demo



http://www.visual-experiments.com/demo/ransac.js/

Application: Cleaning up noisy correspondences!



Panoramic Example

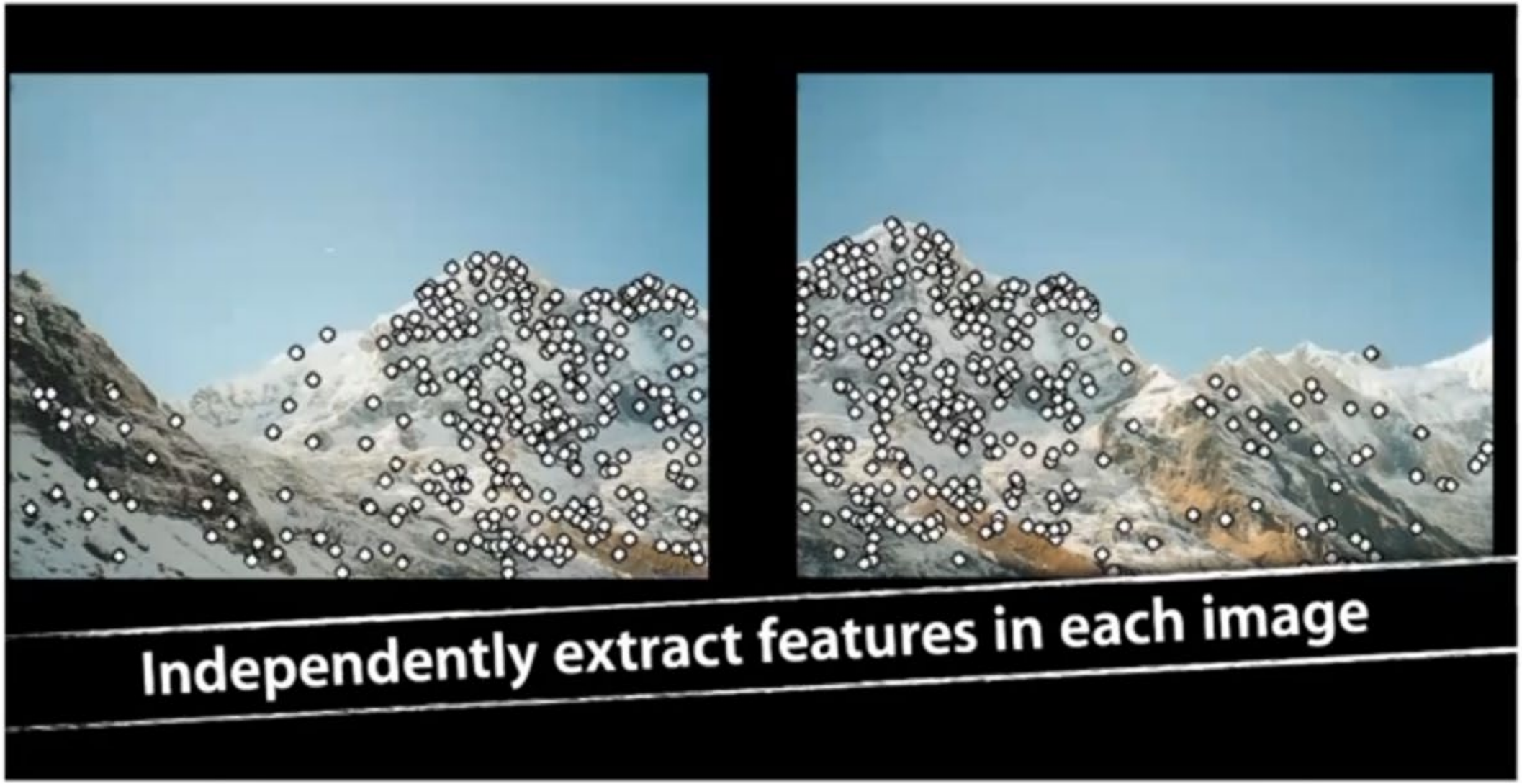
virtual wide-angle camera

Efros

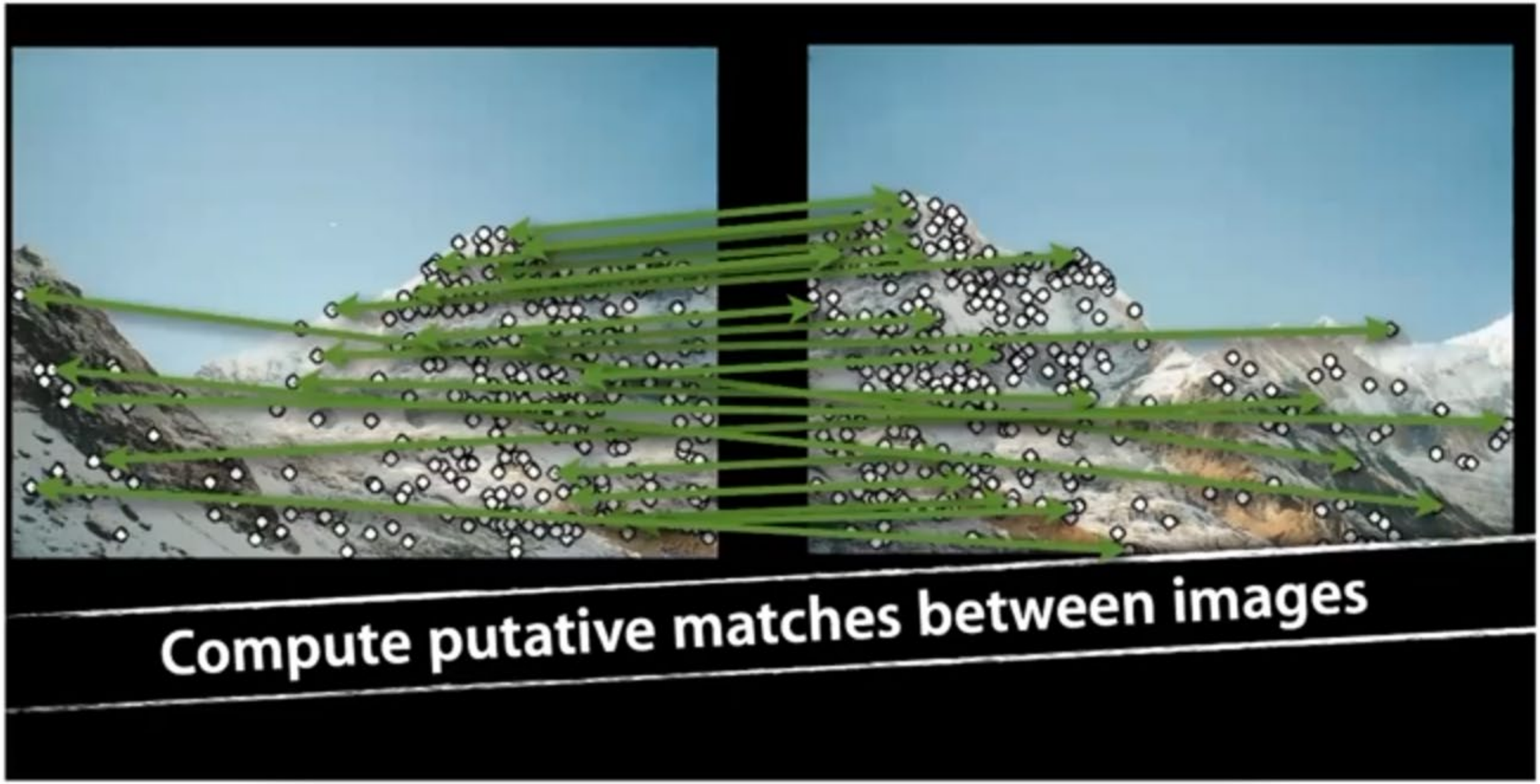
Mosaic Example

Slides from Kosta Derp

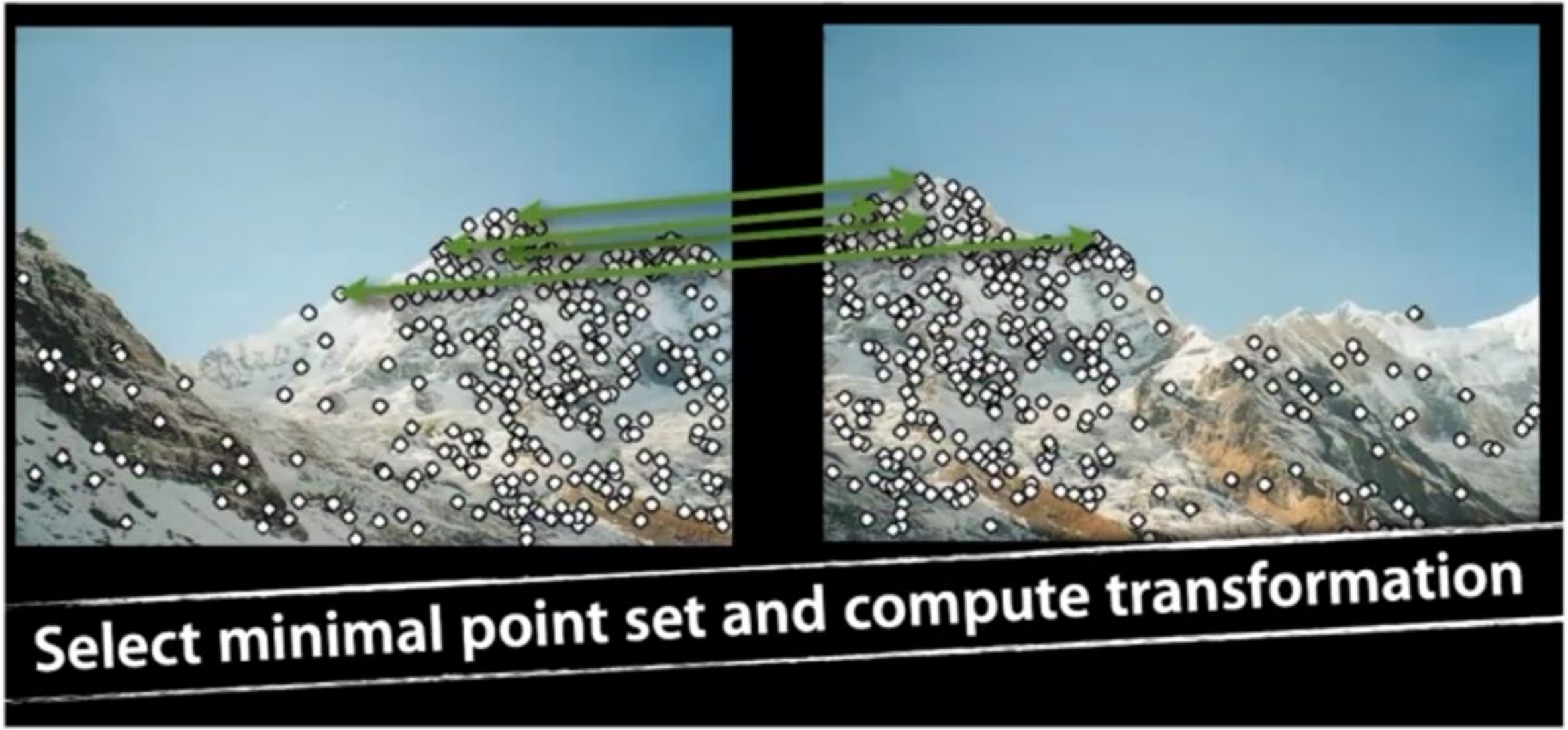
Step 1: Feature Extraction



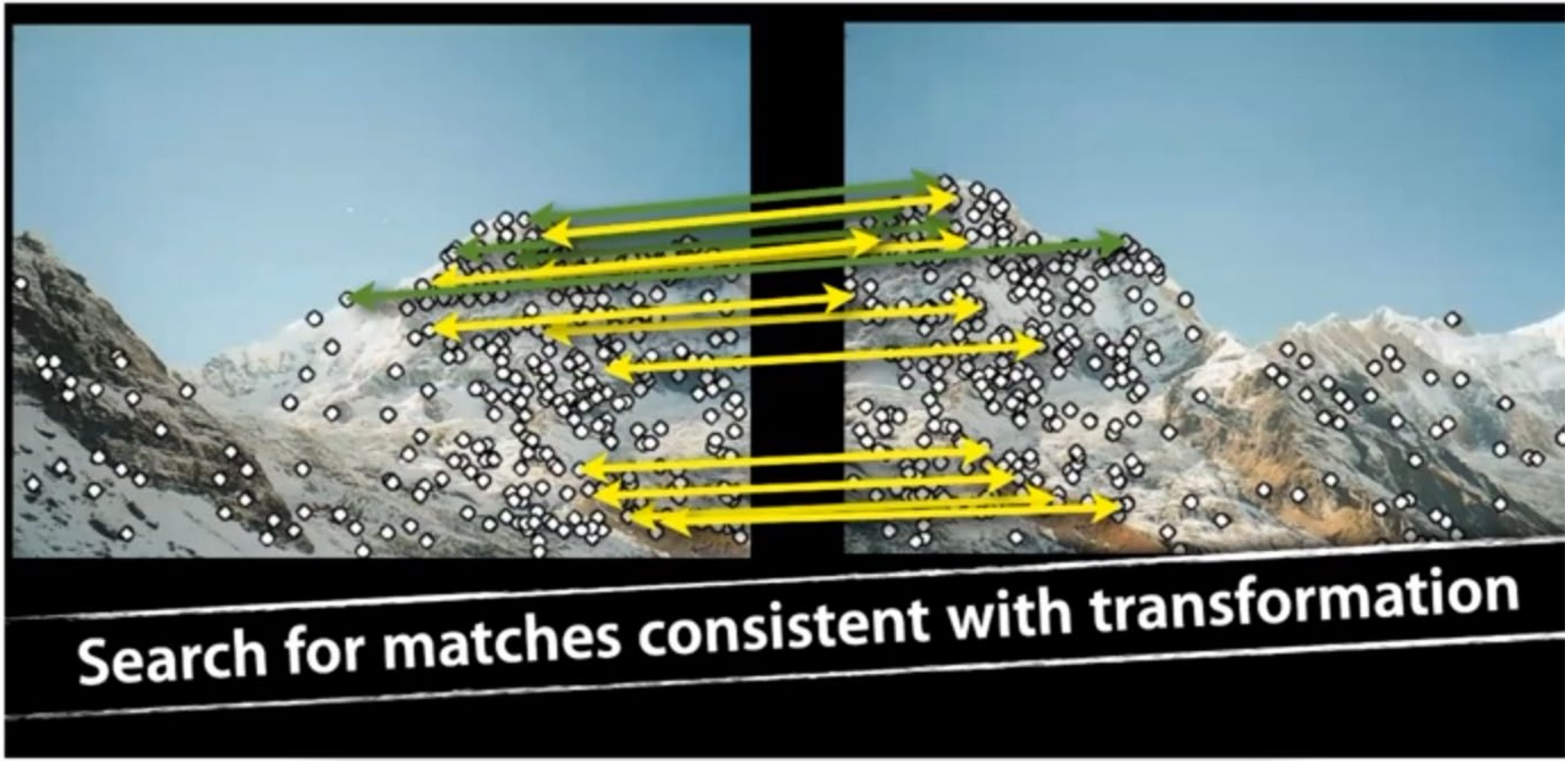
Step 2: Feature Matching



Step 3: Minimal Point Set for Estimation

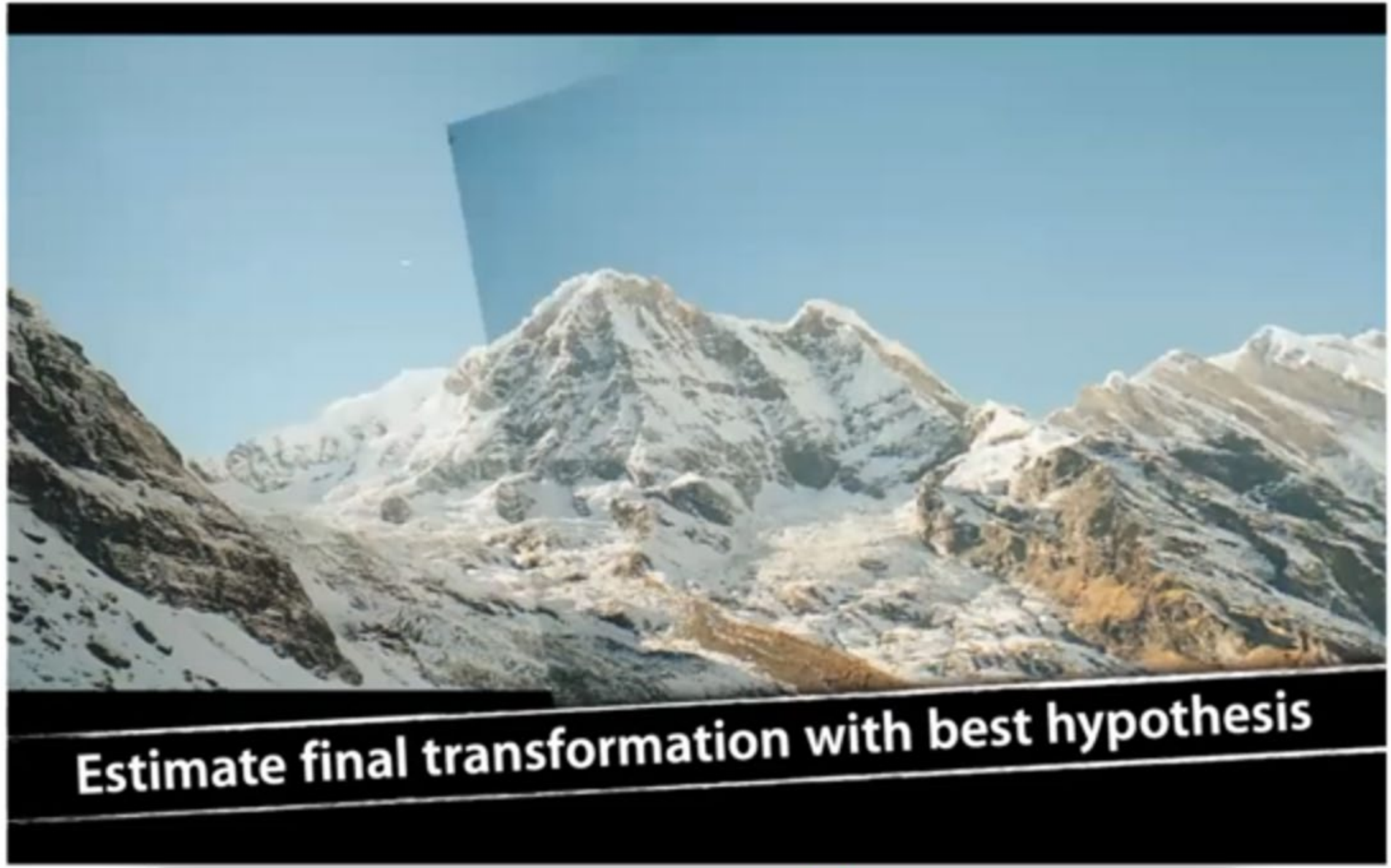


Step 4: Check Consistency



Repeat Steps 3-4 to select best parameters

Check Warping Results



RANSAC vs. Hough

- RANSAC is more efficient when fraction of outliers is low
- Hough is intractable for large number of unknowns

RANSAC is a workhorse, widely used in a large number of practical settings today, beyond just computer vision.

What methods for what types of correspondences?

- 3D-> 3D correspondences
- 3D-> 2D correspondences
- 2D->2D correspondences

- Procrustes to solve R, T
- Pose from Point correspondences (PnP / P3P)
- Structure from Motion (SfM) and /or image stitching

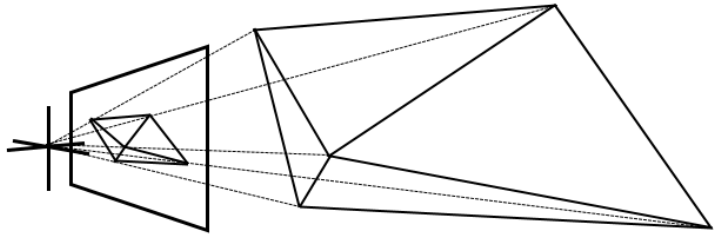
Correspondences are really key! But where do they actually come from?

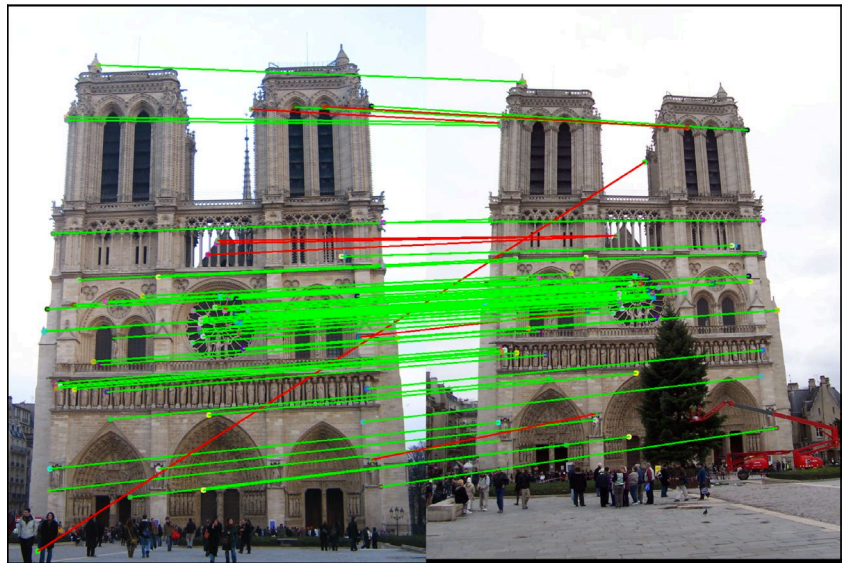
One solution: “optical flow”

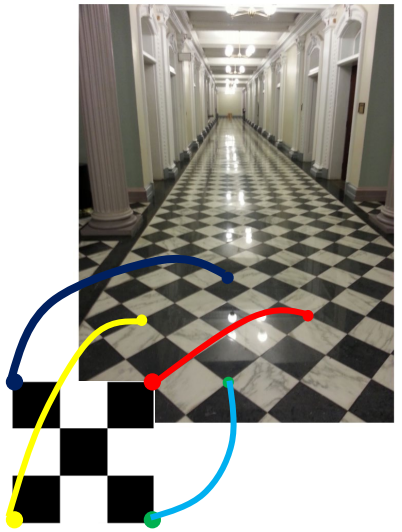
Where Do Point Correspondences Come From?

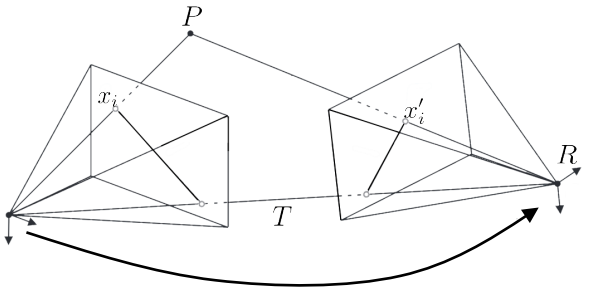
In all the algorithms we have covered thus far we have simply assumed correspondences were given – but how do we compute them?

One solution: “optical flow”









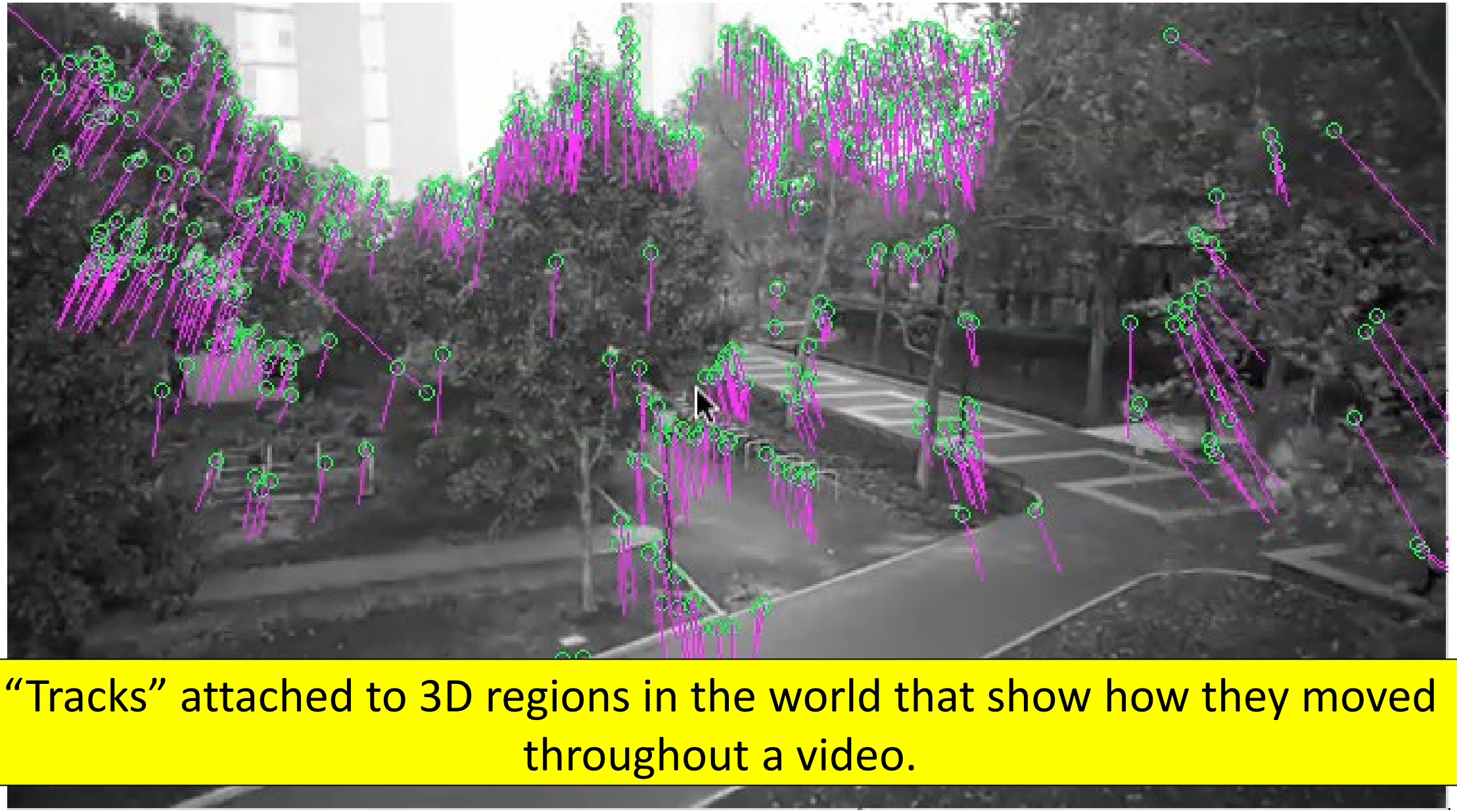
Point Correspondences Through Optical Flow

Optical Flow: Definition

The pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene.

- Ideally, the optical flow is the projection of the 3-D velocity vectors on the image. i.e., the “motion field”.
  - But this won’t always be possible, as we will see.

Target Optical Flow Output

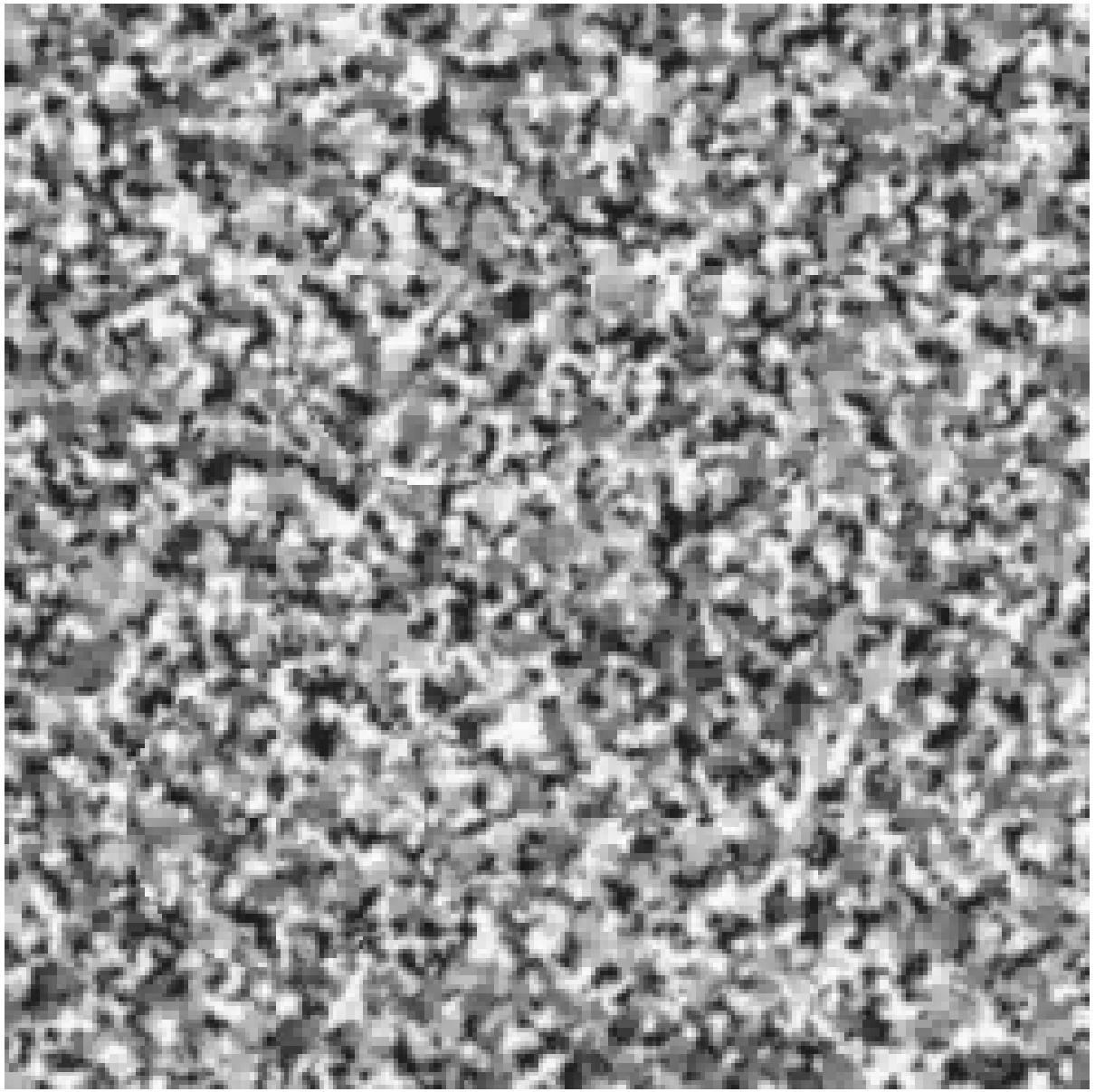


"Tracks" attached to 3D regions in the world that show how they moved throughout a video.

Optical Flow is Useful Beyond Geometric Vision

- Independently interesting! For example, for video compression.
  - Rather than encoding every patch of the second frame from scratch, you could make your job much easier by saying that patch no. k of frame 2 is the same as patch no. j of frame 1, for example.
    - Large gains (often 2-3 orders of magnitude!) in storage, transmission etc.
- When flow is computed over video frames from a moving camera in a (mostly) static world, it yields correspondences that can be used e.g. in SfM.
- We want efficient methods particularly for real-time use cases, like on a robot.

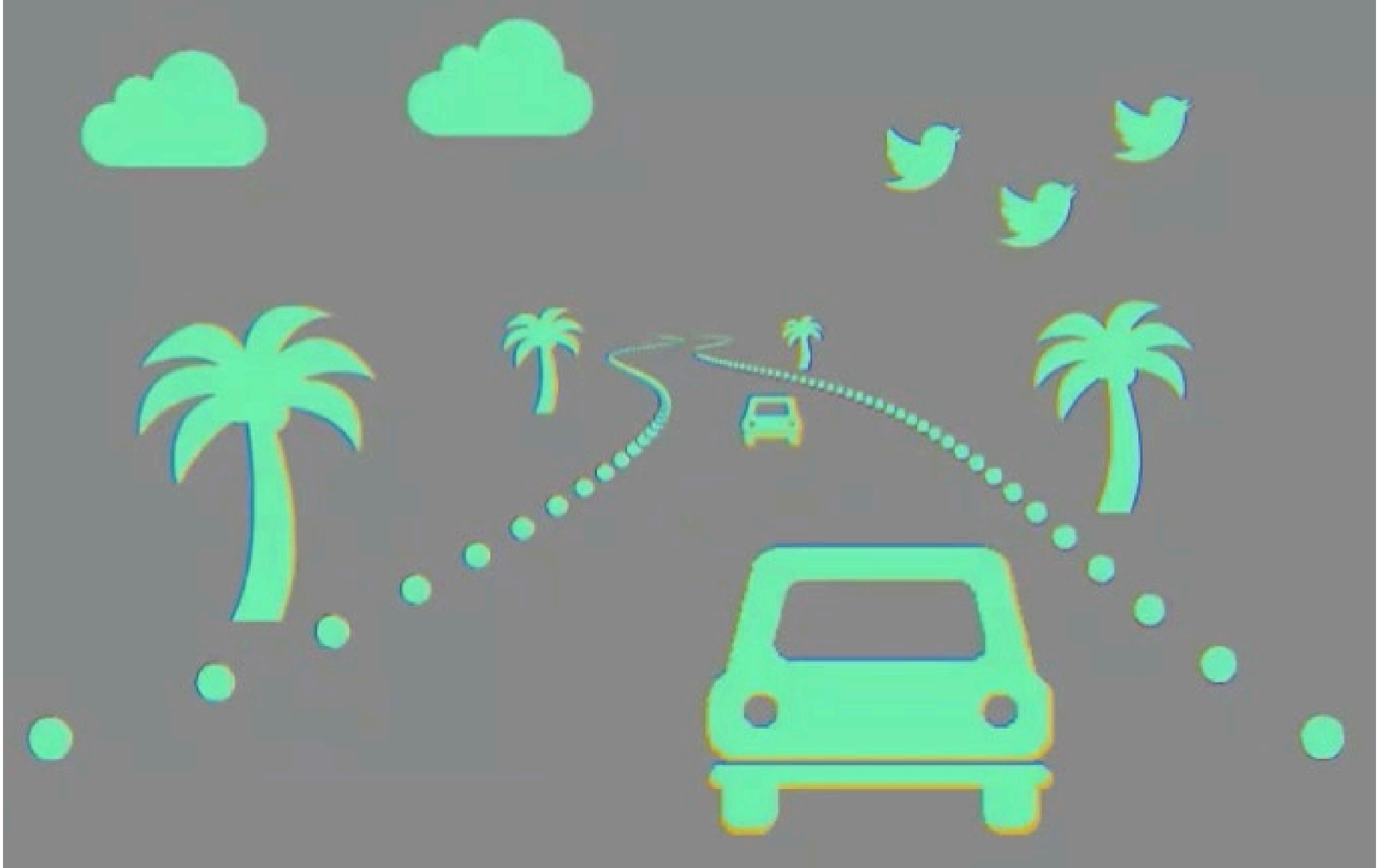
Motion is a strong segmentation cue

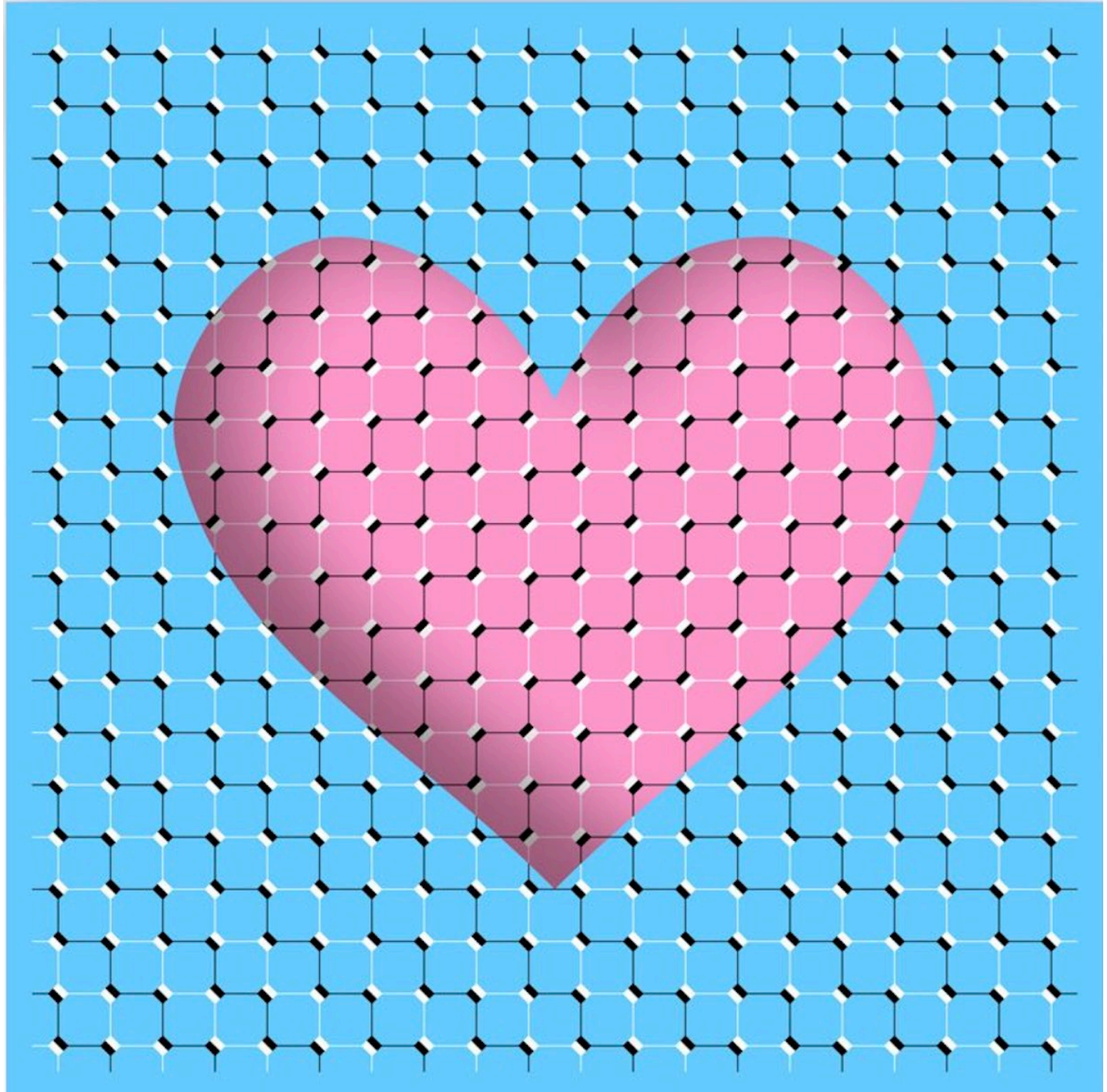


“Segmentation”: grouping pixels into objects, object parts etc.

Animals are extremely reliant on detecting motion in natural scenes for segmentation and other perception objectives. But this is not an easy problem.

By constructing various non-natural scenes, it is possible to break our motion detection!





Illusory motions!

Often happen because the eye is constantly roving the scene, and we are having to constantly compensate for the eye's own motion when figuring out what else is moving in the scene.

Image Motion

To try and solve this problem let's consider two consecutive frames in a video

$I_t(x, y)$

$I_{t+1}(x, y)$

Can you tell how the camera is moving?

Not so easy, but let's flip between the two images to make it easier

Flipbook Slide

Image I_t

Flipping between these slides, we see we could move each pixel in the first to match the second. The motion of each pixel in this is called optical flow

Flipbook Slide

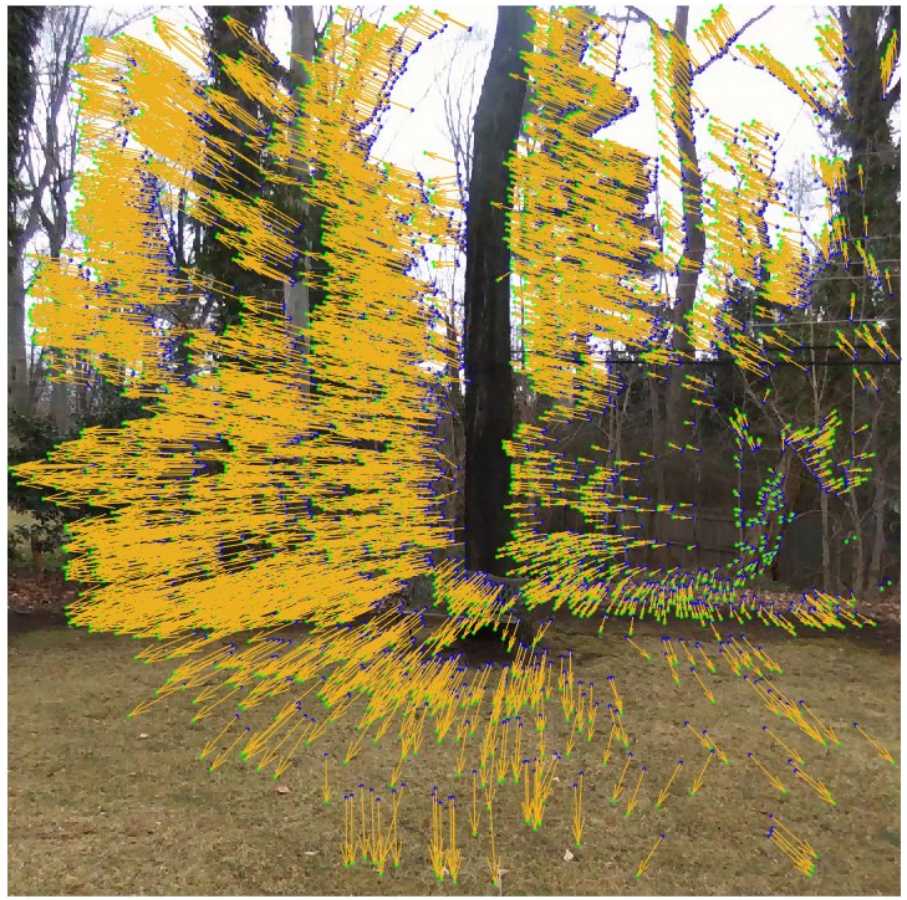
Image I_{t+1}

Flipping between these slides, we see we could move each pixel in the first to match the second. The motion of each pixel in this is called optical flow

Target Output of Optical Flow

For a pixel in the first image, optical flow should tell us which pixel in the second image it matches with, i.e., “corresponds” to.

$$I_t(x, y) \longrightarrow I_{t+1}(x, y)$$



Note that we are not showing correspondences for *every* pixel, back to that in a few slides.

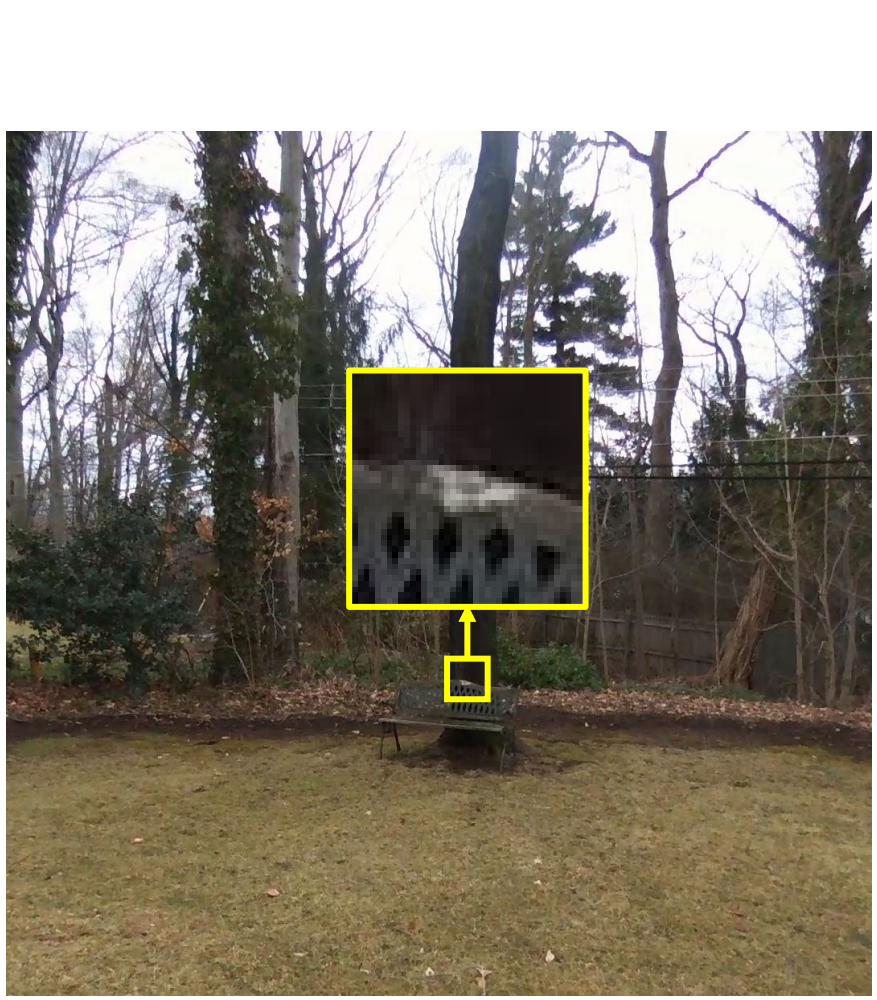
Problem: Individual pixels can be very ambiguous to match across images.

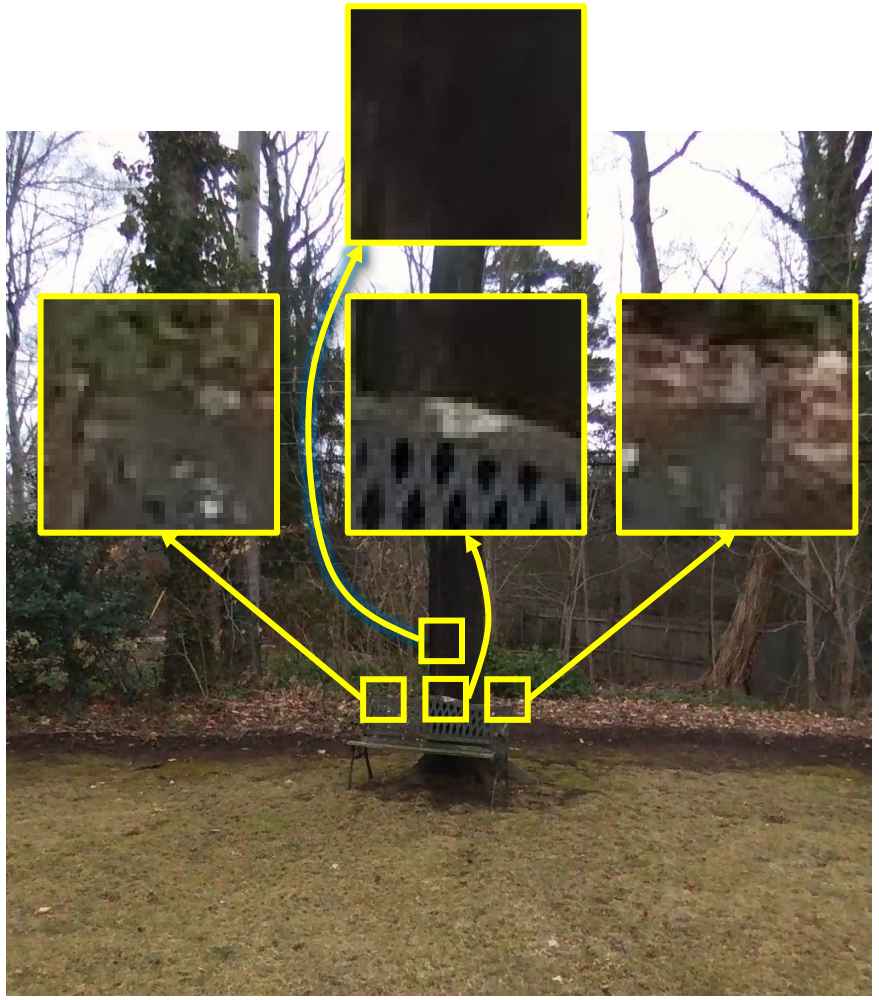
(A more precise version of this soon)

Optical Flow Over Patches

Solution: Let's match small local regions ("patches") between the two images.

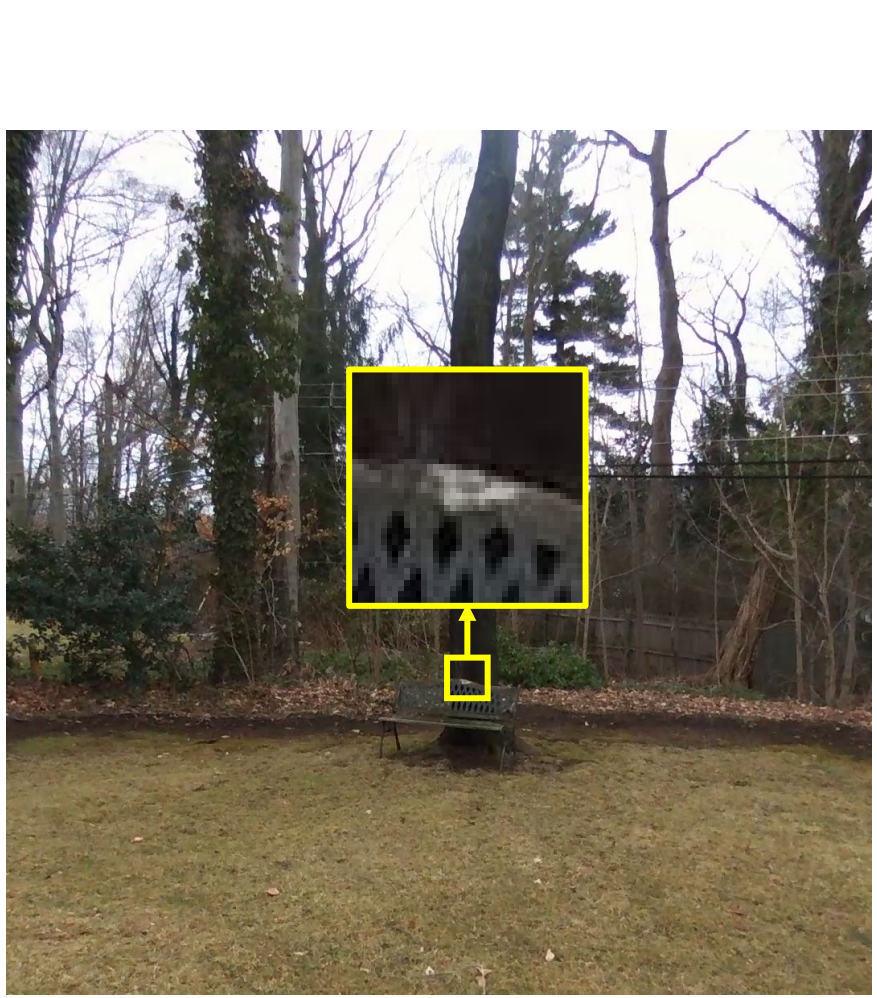
(Assumption: All pixels within a patch move in the same way)

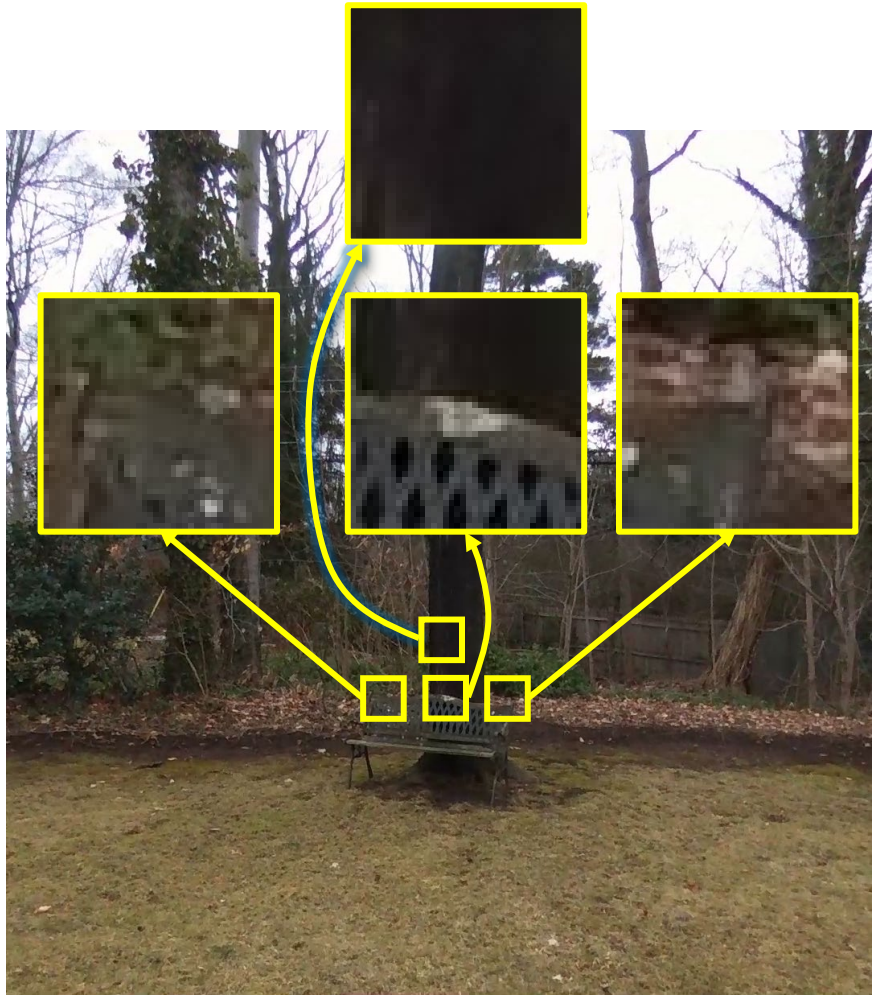




Optical Flow Over Patches

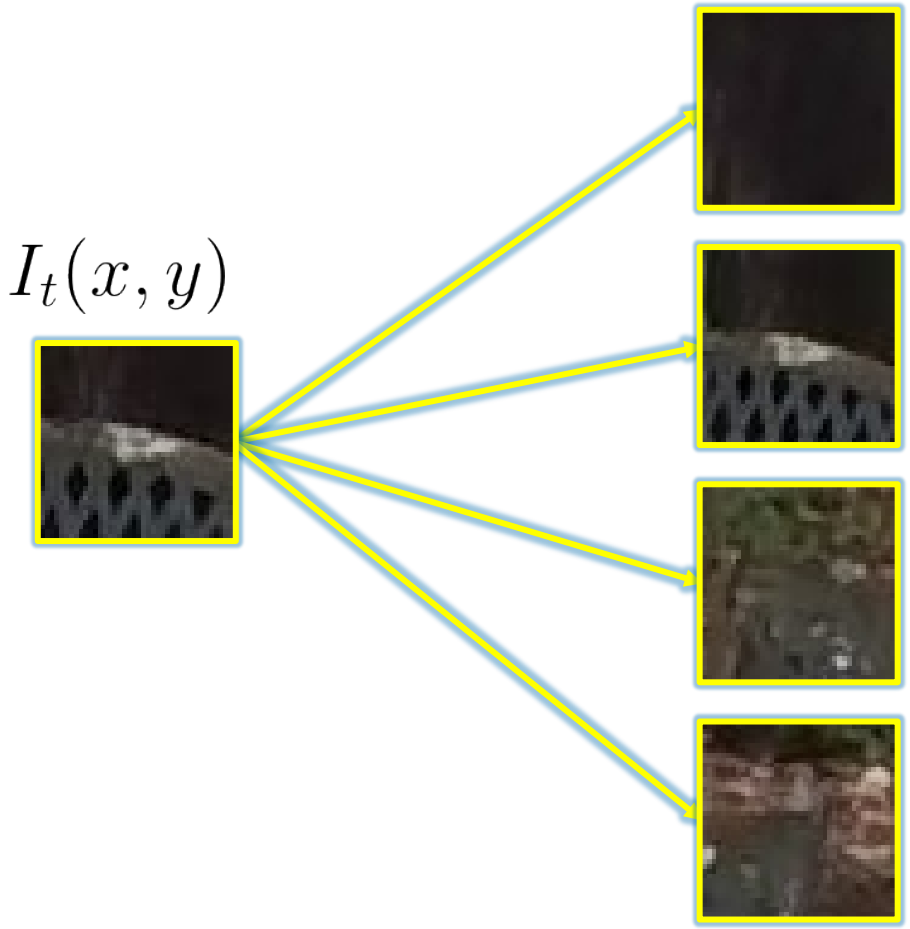
Let’s focus first on image patches that we can locally distinguish from other patches





Optical Flow as Local Search

If the video frames are close enough we can look for these salient patches in nearby images!



Which nearby patch in the next image is the closest?

Side note: Some people use the reverse convention in terms of which image to search on, but the math is the same

Local Search Objective Function

(x_0, y_0) Pixel where we are computing the optical flow

$\mathcal{N}$ Neighborhood patch around a pixel

$(\delta x, \delta y)$ Offset we are optimizing for – the optical flow

The function we are trying to optimize is (assuming a continuous image):

$$(\delta x, \delta y) = \underset{\delta x, \delta y}{\operatorname{argmin}} \underbrace{\int \int_{(x,y)\in\mathcal{N}(x_0,y_0)}} I_t(x,y) - I_{t+1}(x+\delta x, y+\delta y)\, dxdy$$

If we discretize:

$$\underset{\delta x, \delta y}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{N}(x_0,y_0)} (I_t(x,y) - I_{t+1}(x+\delta x, y+\delta y))^2$$

Exhaustive search over all possible sub-pixel motions $\delta x, \delta y$? Intractable!

Local Search Simplification

To speed things up, assume that the change is small

$$I_t(x,y) - I_{t+1}(x+\delta x, y+\delta y)$$

$$\approx I_t(x,y) - \left(I_{t+1}(x,y) + \nabla I_{t+1}(x,y)^T \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \right)$$

Taylor expansion

$$= \Delta I_t(x,y) - \nabla I_{t+1}(x,y)^T \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

Note the 2 different triangles

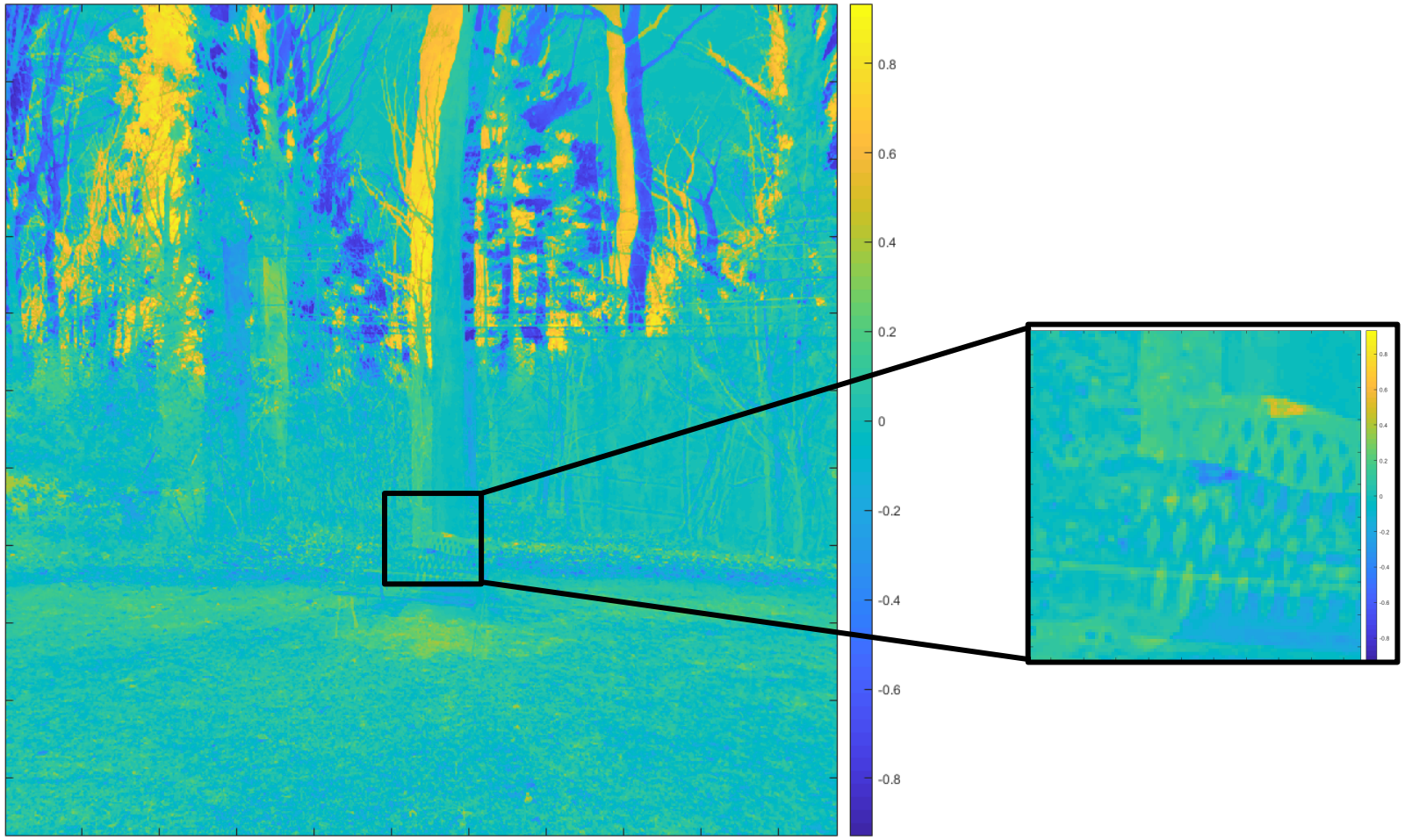
We want to select $\delta x, \delta y$ to minimize the sum of the square of this quantity over a neighborhood.

First, let's understand its two terms, the difference image, and the gradient.

$$\underset{\delta x, \delta y}{\operatorname{argmin}} \sum_{(x,y) \in \mathcal{N}(x_0,y_0)} (I_t(x,y) - I_{t+1}(x+\delta x, y+\delta y))^2$$

Term 1: Difference Image

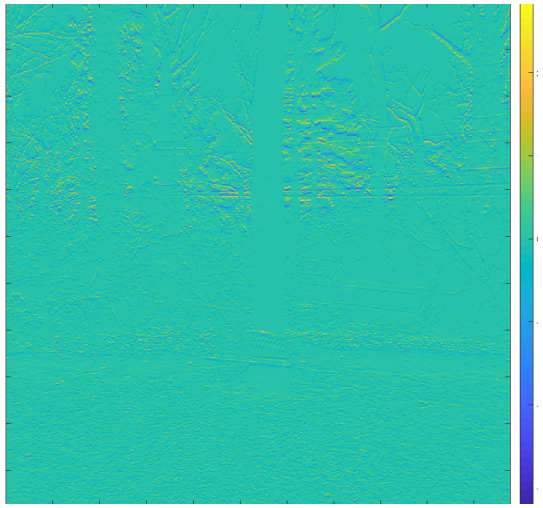
$$= \boxed{\Delta I_t(x,y)} - \nabla I_{t+1}(x,y)^T \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$



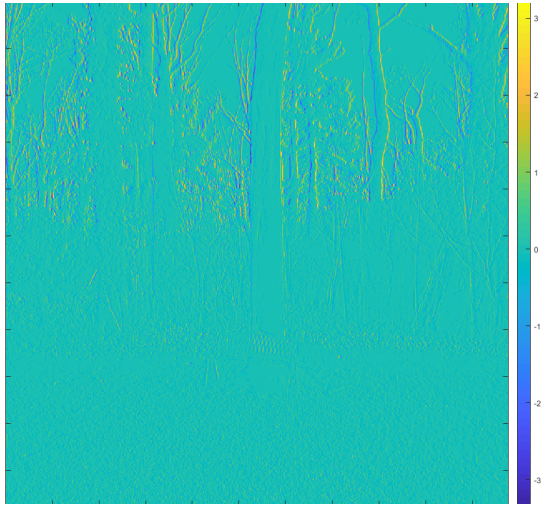
Term 2: Spatial Gradient

$$= \Delta I_t(x,y) - \boxed{\nabla I_{t+1}(x,y)^T} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$

These gradients are easy to compute efficiently using "convolutions". More on that later in the course.

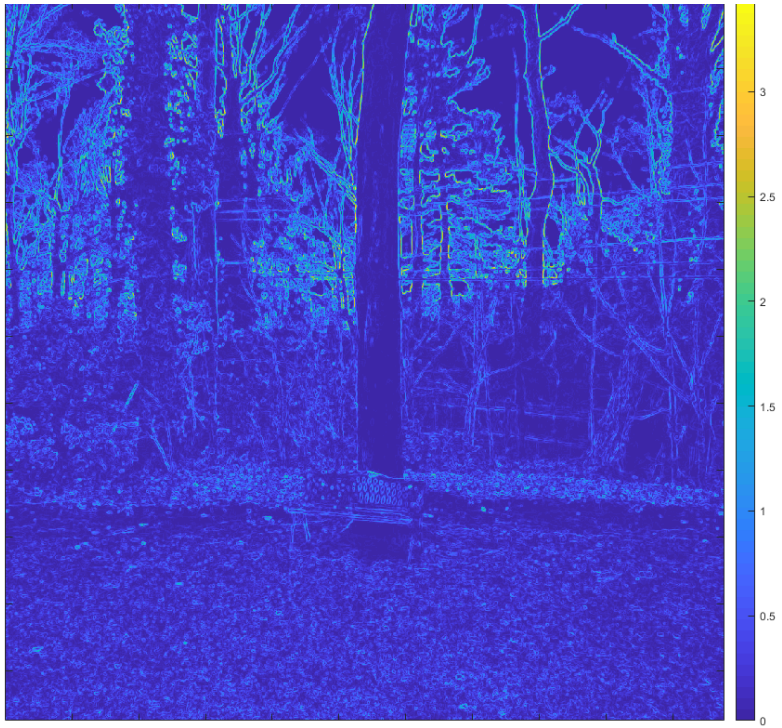


$\nabla_x I_{t+1}(x,y)$



$\nabla_y I_{t+1}(x,y)$

$\|\nabla I_{t+1}(x,y)\|$



Back to Optimizing Flow

The new cost function becomes:

$$(\delta x, \delta y) = \underset{\delta x, \delta y}{\operatorname{argmin}} \sum_{x,y \in \mathcal{N}(x_0,y_0)} \left(\Delta I_t(x,y) - \nabla I_{t+1}(x,y)^T \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} \right)^2$$

This is the type of linear least squares problem we regularly solve, when faced with overdetermined systems of linear equations that look like:

$$\left[\nabla I_{t+1}(x,y)^T \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \Delta I_t(x,y) \right]_{x,y \in \mathcal{N}(x_0,y_0)}$$

Q: What if we had defined a single-pixel "patch" $\mathcal{N}$, i.e., $\mathcal{N}(x_0, y_0) = \{(x_0, y_0)\}$?

A: One equation, two unknowns. Unsolvable! This is the "aperture problem" of optical flow.

You need a neighborhood to compute flow.

Lucas-Kanade Optical Flow

$$\left[\nabla I_{t+1}(x,y)^T \begin{bmatrix}\delta x\\ \delta y\end{bmatrix} = \Delta I_t(x,y)\right]_{x,y\in\mathcal{N}(x_0,y_0)}$$

Stacking the equations:

x derivative

y derivative

$$\begin{bmatrix} \nabla_x I\big|_{p_0} & \nabla_y I\big|_{p_0} \\ \vdots & \vdots \\ \nabla_x I\big|_{p_i} & \nabla_y I\big|_{p_i} \\ \vdots & \vdots \\ \nabla_x I\big|_{p_n} & \nabla_y I\big|_{p_n} \end{bmatrix} \begin{bmatrix}\delta x\\ \delta y\end{bmatrix} = \begin{bmatrix} \Delta I\big|_{p_0} \\ \vdots \\ \Delta I\big|_{p_i} \\ \vdots \\ \Delta I\big|_{p_n} \end{bmatrix}$$

A $\quad$ $\boldsymbol{x}$ $\quad$ b

At point p_i in the patch

Spatial gradients ∇I computed on second image I_{t+1}

Pixel differences computed as $\Delta I = I_t - I_{t+1}$

$$x^* = (A^T A)^{-1} A^T b$$

Recall that for overdetermined $Ax = b$, we solve $\min_x ||Ax - b||_2^2$ using the pseudo-inverse $(A^T A)^{-1} A^T b$

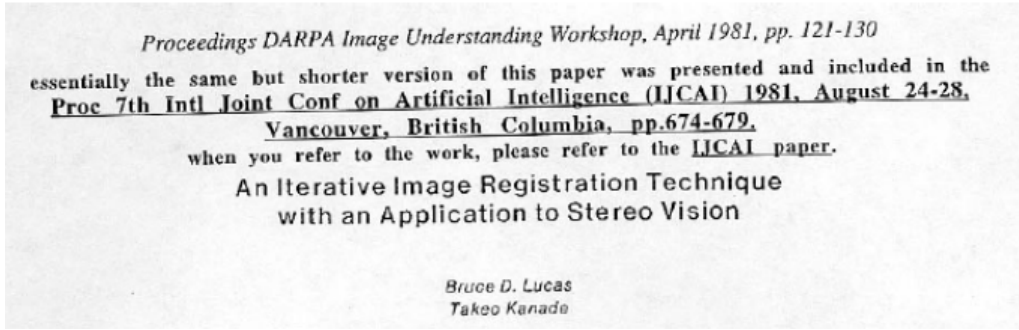
Lucas-Kanade (LK) Optical Flow

- If you explicitly plug A from the last slide into $(A^TA)^{-1}A^Tb$, you get:

$$\begin{pmatrix}\delta x^* \\ \delta y^*\end{pmatrix} = \left(\sum_{(x,y)} \nabla I_{t+1}(x,y)\nabla I_{t+1}(x,y)^T\right)^{-1} \left(\sum_{(x,y)} \Delta I_t(x,y)\nabla I_{t+1}(x,y)\right)$$

(But easier to just remember the linear system and pseudoinverse)

Invented in 1981, still very widely used!

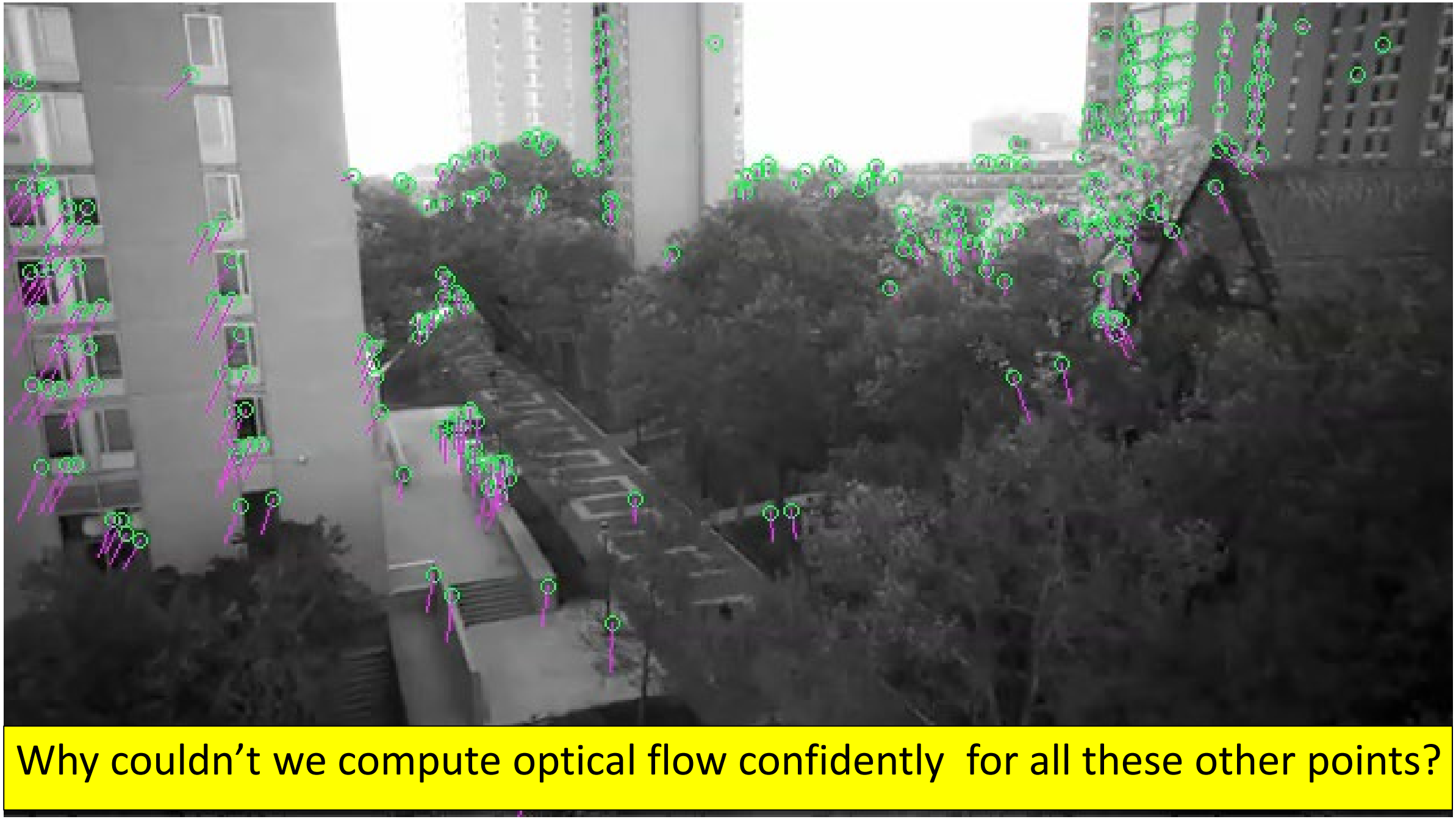


Proceedings DARPA Image Understanding Workshop, April 1981, pp. 121-130

essentially the same but shorter version of this paper was presented and included in the Proc 7th Intl Joint Conf on Artificial Intelligence (IJCAI) 1981, August 24-28, Vancouver, British Columbia, pp.674-679.
when you refer to the work, please refer to the IJCAI paper.

An Iterative Image Registration Technique
with an Application to Stereo Vision

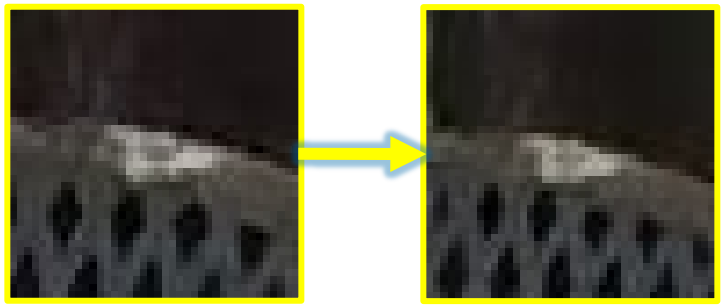
Bruce D. Lucas
Takeo Kanade



Why couldn’t we compute optical flow confidently for all these other points?

Assumptions We've Made: Brightness Constancy

Brightness Constancy: Color doesn't change between different viewpoints (based on "Lambertian surface reflectance" assumption)



Counterexample

$$\sum_{(x,y)\in\mathcal{N}(x_0,y_0)} (I_{t+1}(x,y) - I_{t+1}(x+\delta x, y+\delta y))^2$$

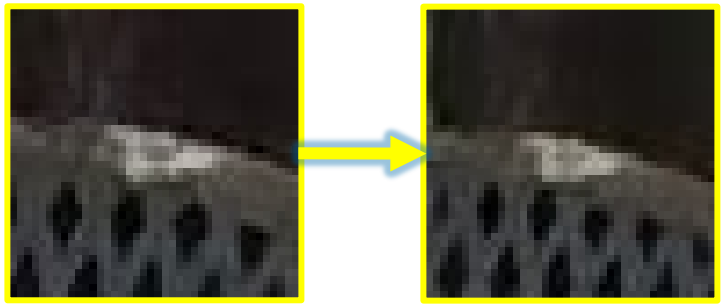
Assumptions We've Made: No Occlusions

No occlusions: things stay in sight

Counterexample

$$\sum_{(x,y)\in\mathcal{N}(x_0,y_0)} (I_{t+1}(x,y) - I_{t+1}(x+\delta x, y+\delta y))^2$$

Assumptions We've Made



Minimal geometric deformations:
No large rotations or scaling

Minimal patch displacement: Taylor expansions only work for small translations

Could run optical flow over a pyramid.

Counterexample

Counterexample

Assumptions We've Made: Small Motions

Minimal patch displacement: Taylor expansions only work for small translations

Counterexample

Solution: could solve optical flow over a pyramid of varying resolutions

Assumptions We've Made: Locally Uniform Motion

Constancy of pixel motion within a small neighborhood.

Requires mostly translatory motion, no large rotations / scaling etc. Also, no major depth discontinuities.

Counterexample