

Introduction to Computer Vision

Kaveh Fathian

Assistant Professor

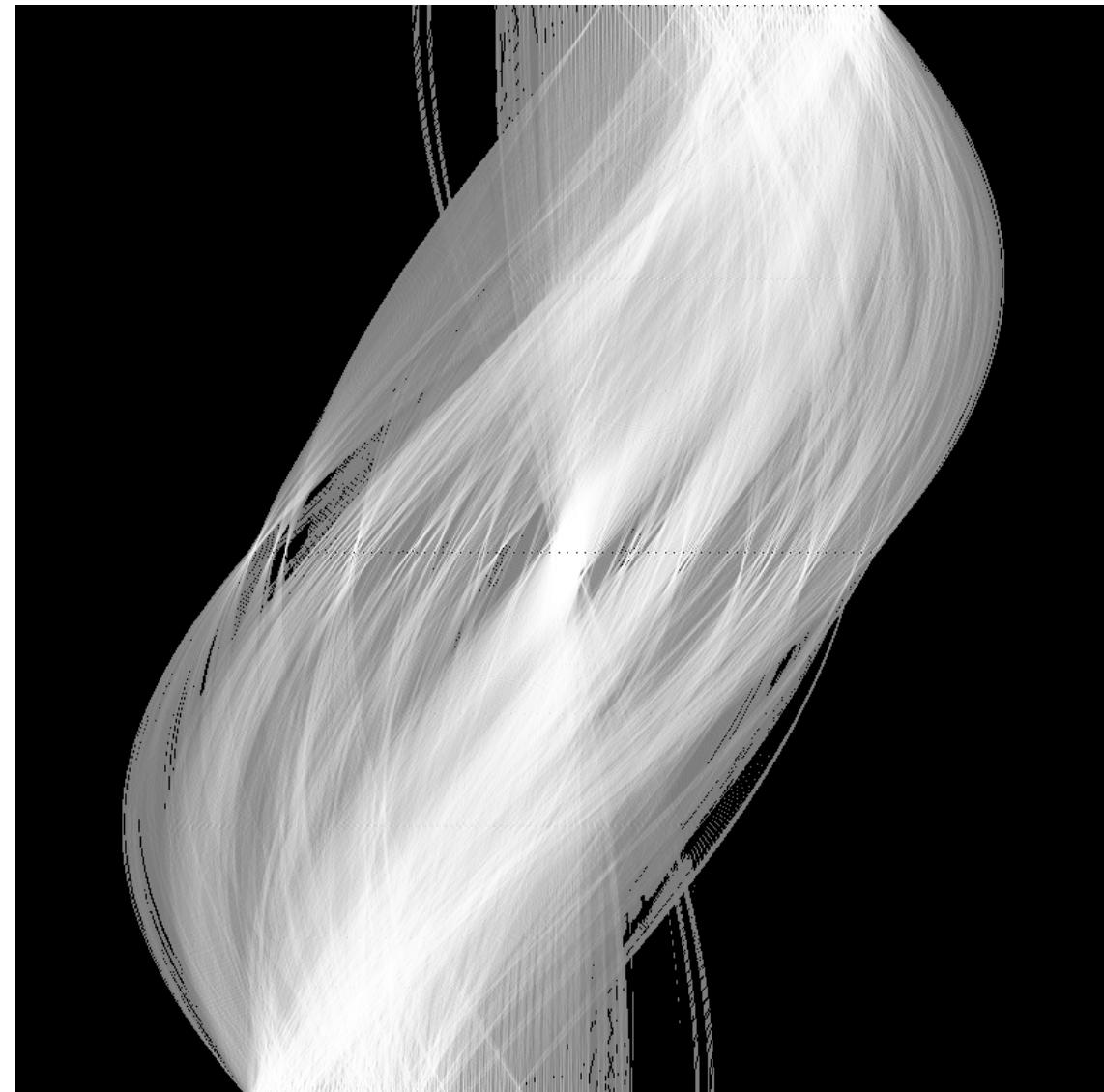
Computer Science Department

Colorado School of Mines

Lecture 7

Learning Outcomes

- Hough Transform
- Line Detection
- Optical Flow



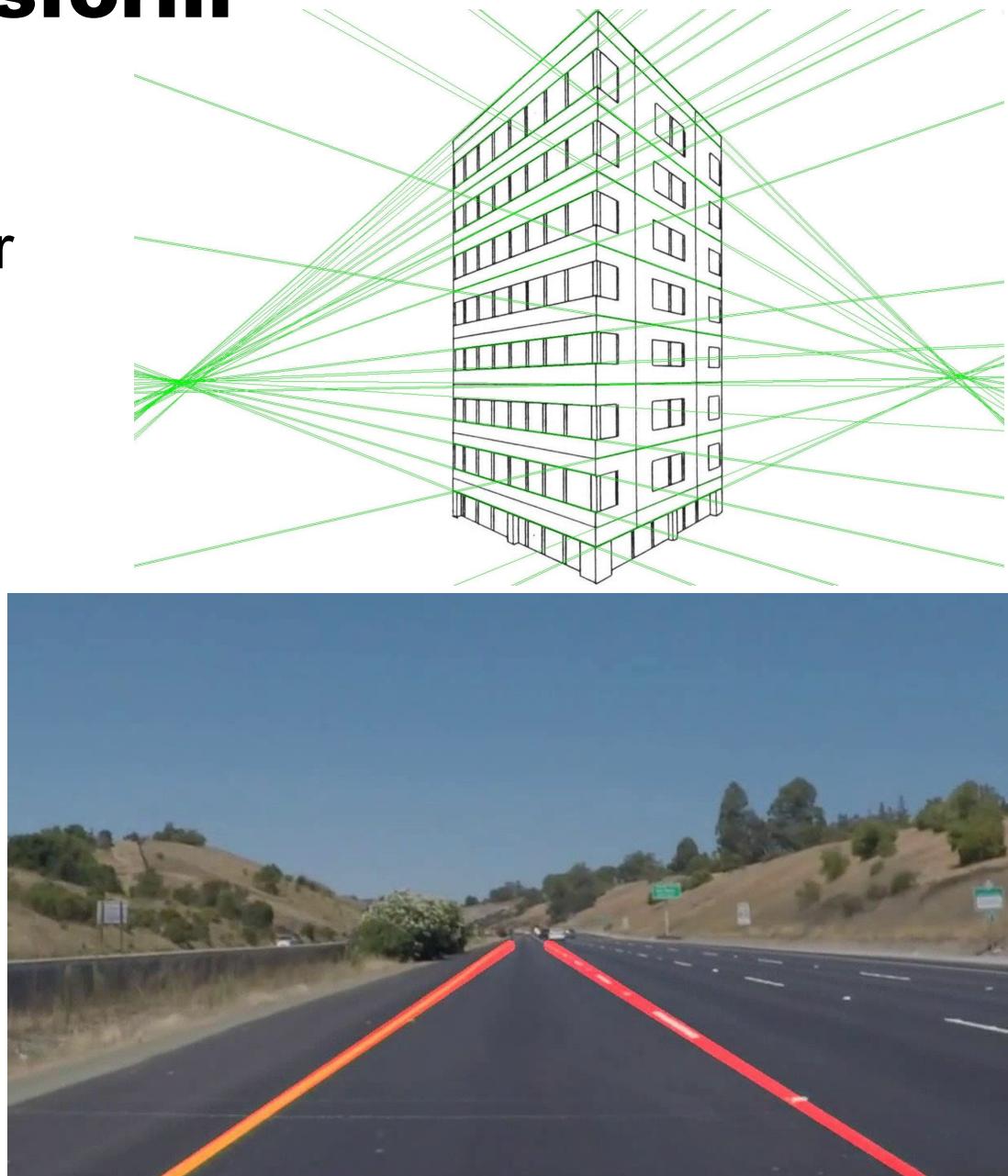
Hough transform

Hough transform is a **generic framework** for detecting a parametric model

Example: Hough transform can detect lines

Why Hough transform? E.g., vs. Canny edge detection?

- Edges don't have to be connected
- Lines can be occluded
- Key idea: edges **vote** for the possible models

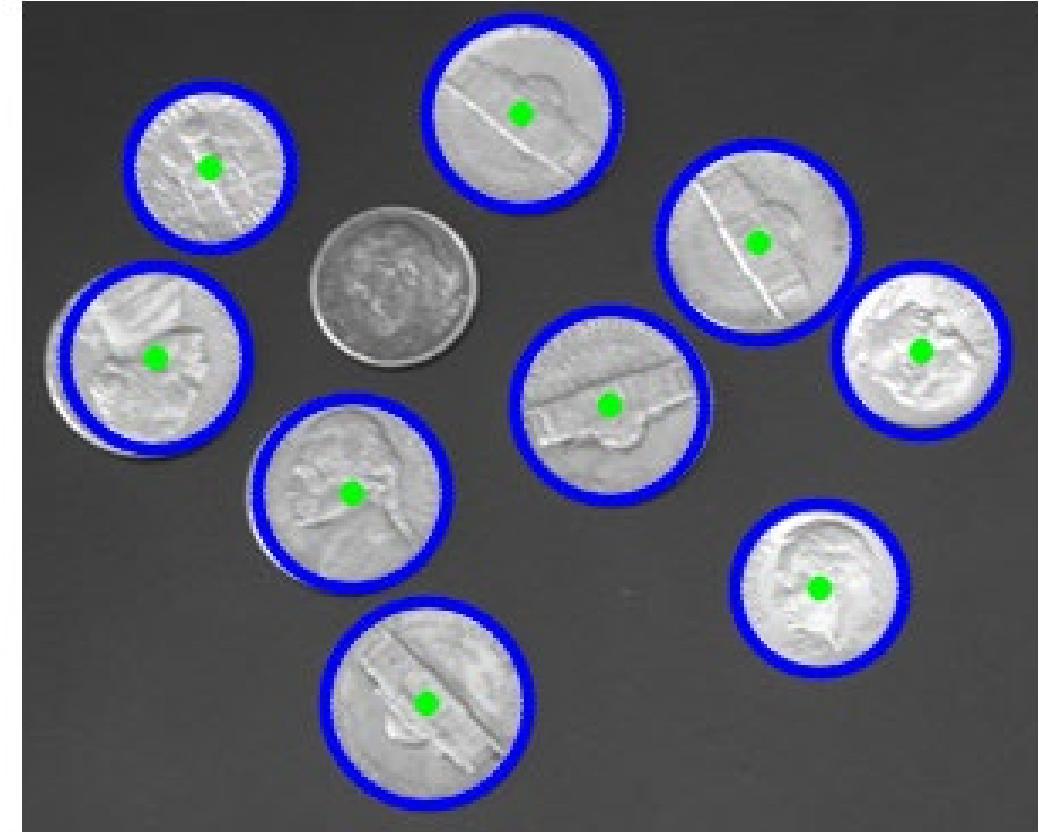


Circles

source



detected circles



Example: Hough transform for circle detection
What is its application?

Image and parameter space

variables
 $y = mx + b$
parameters

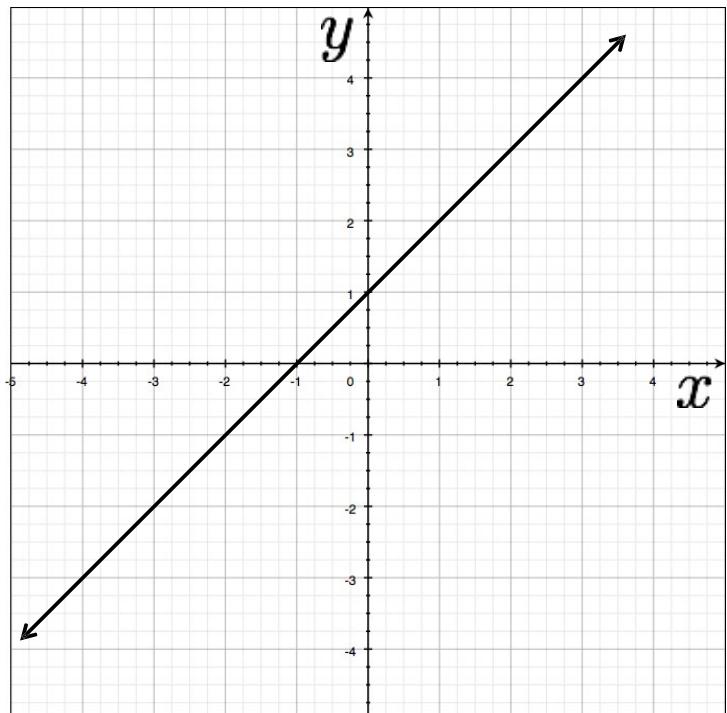
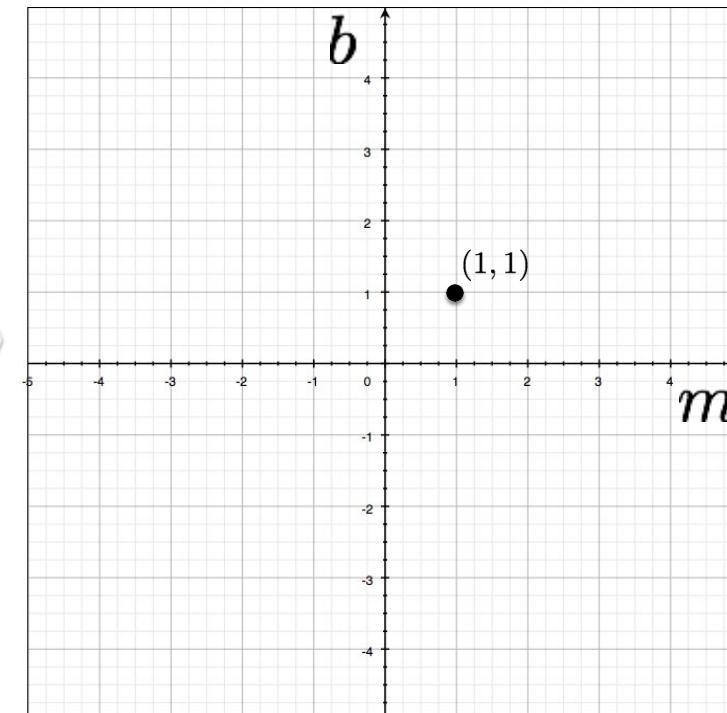


Image space

variables
 $y - mx = b$
parameters

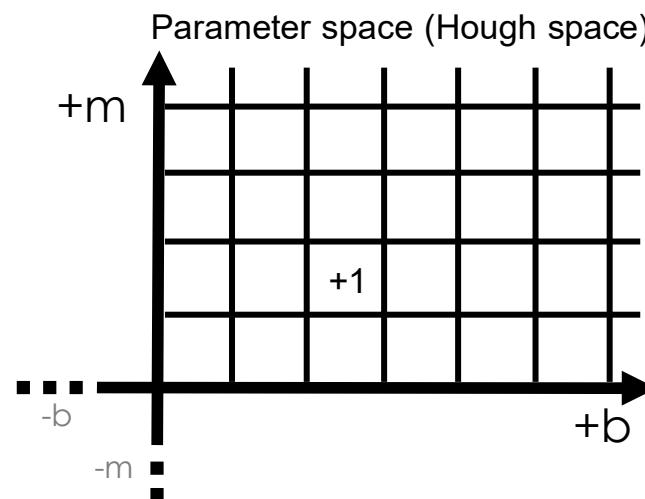
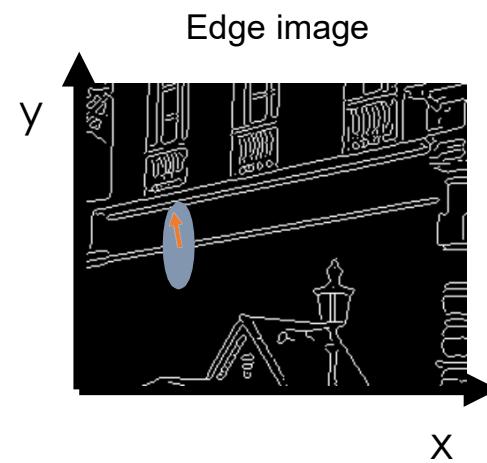


a line
becomes a
point

Parameter space

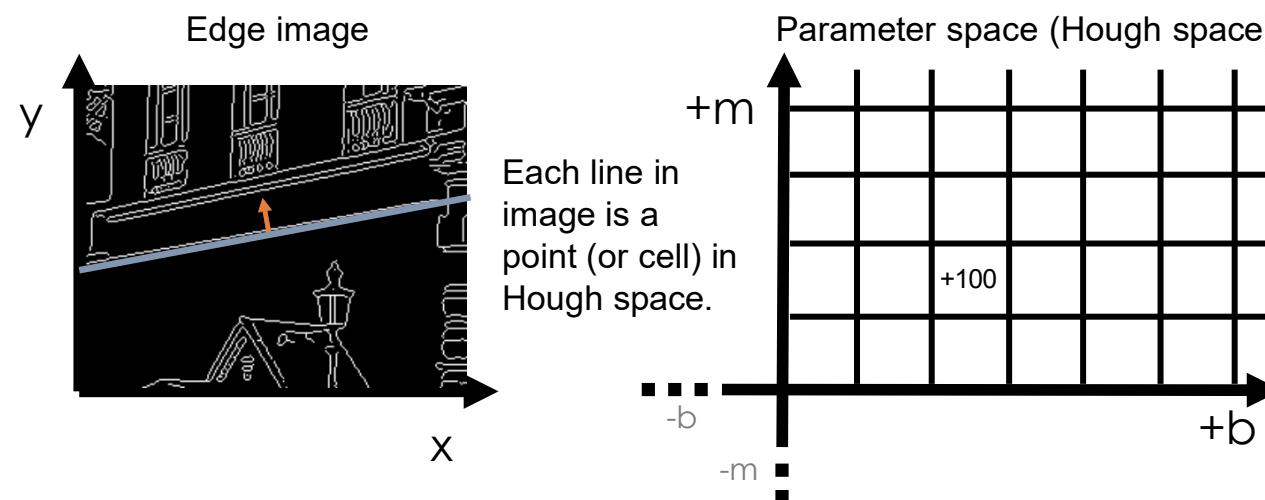
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid



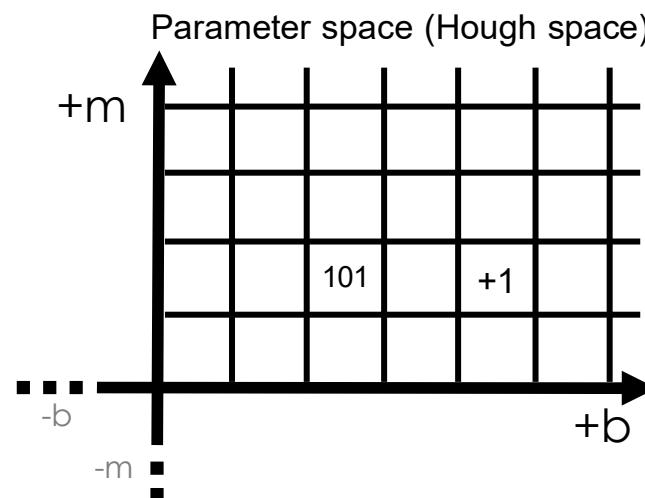
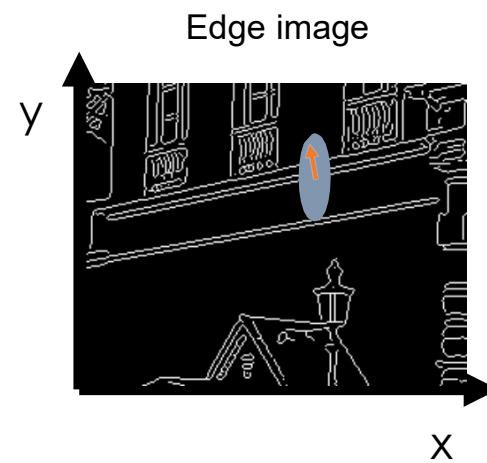
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid



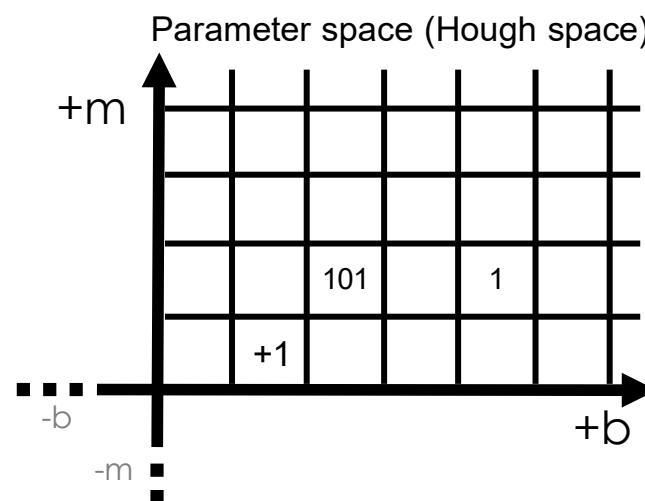
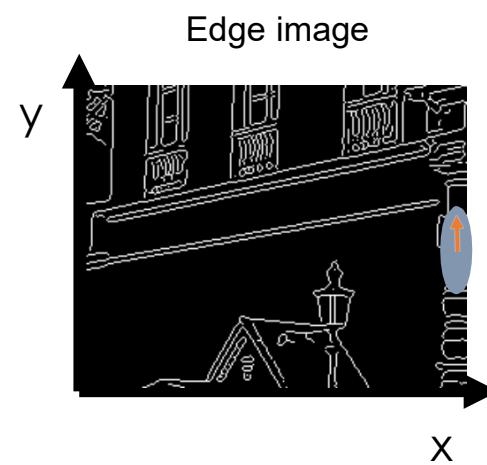
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid



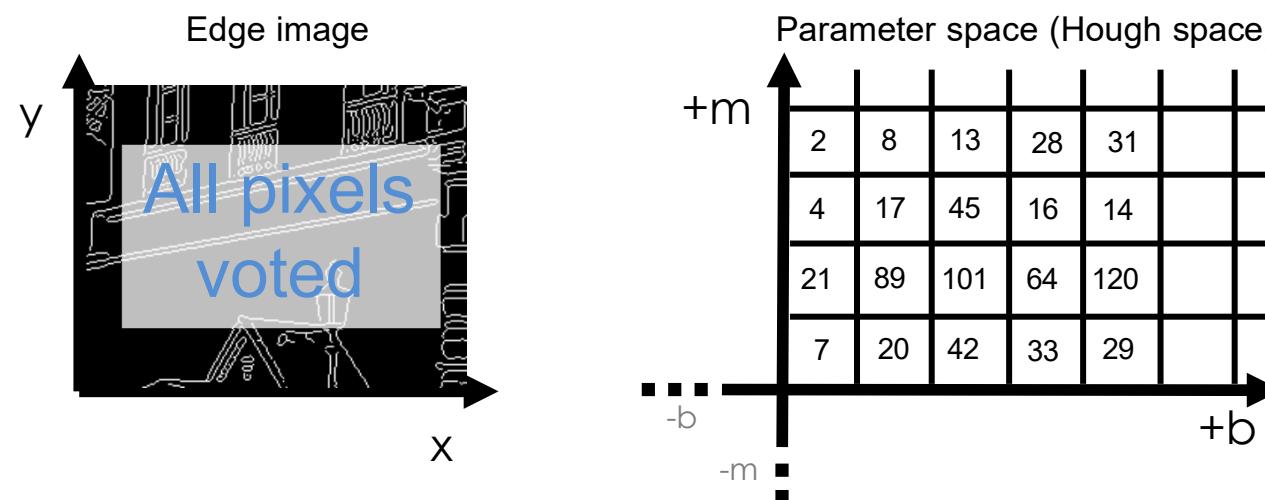
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid



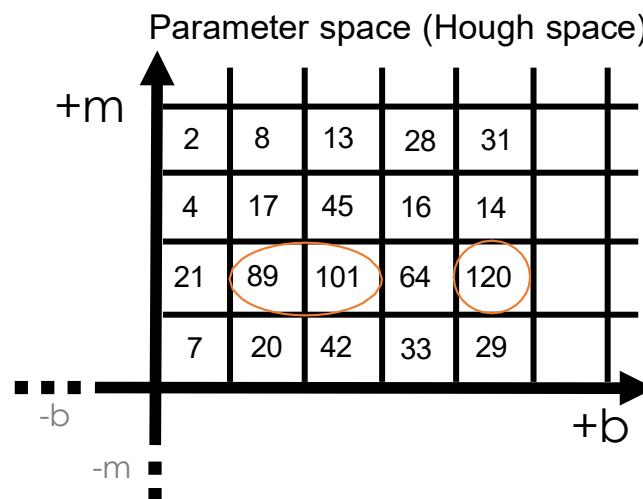
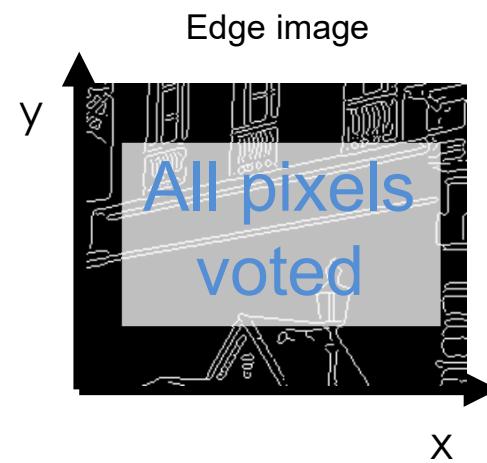
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid



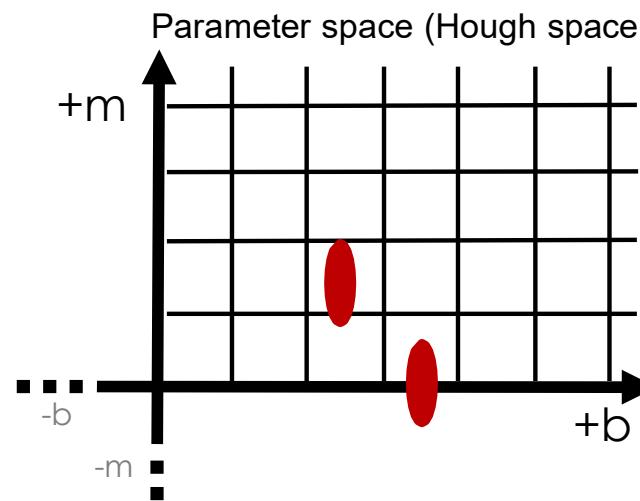
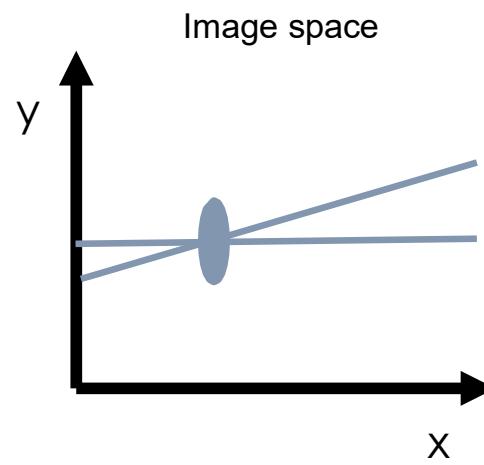
Hough Transform: Outline

- Create a *grid* of candidate m, b parameter values: **accumulator**
 - Why a grid?
 - m, b are continuous; grid discretizes into hypotheses.
- Each edge pixel votes for a set of parameters, which increments those values in grid
- Find maxima – our line candidates.



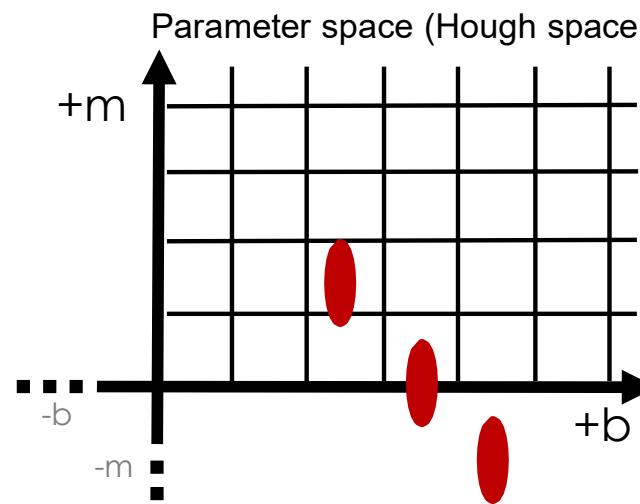
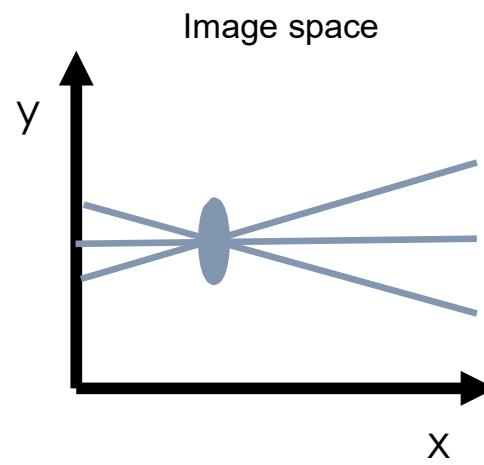
Hough Transform: Step back

- Hough space represents all possible lines.
- With gradient information constraint:
 - Edge is single point in Hough space.
- Without gradient orientation information?



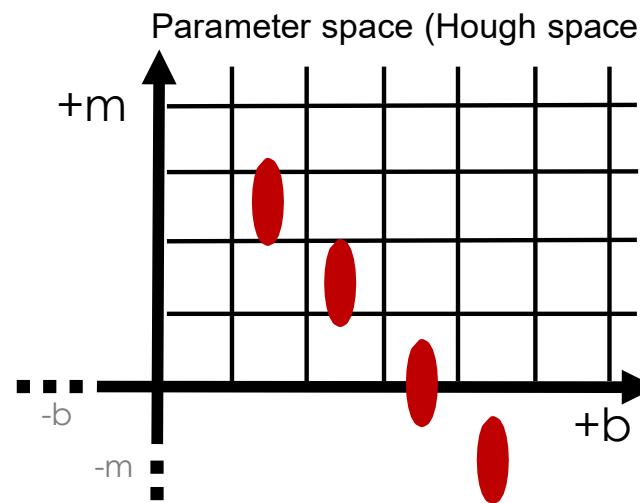
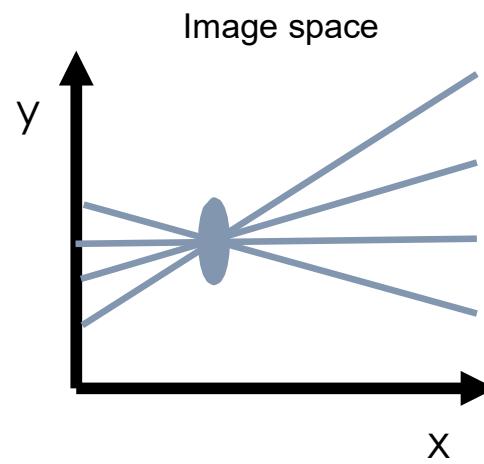
Hough Transform: Step back

- Hough space represents all possible lines.
- With gradient information constraint:
 - Edge is single point in Hough space.
- Without gradient orientation information?



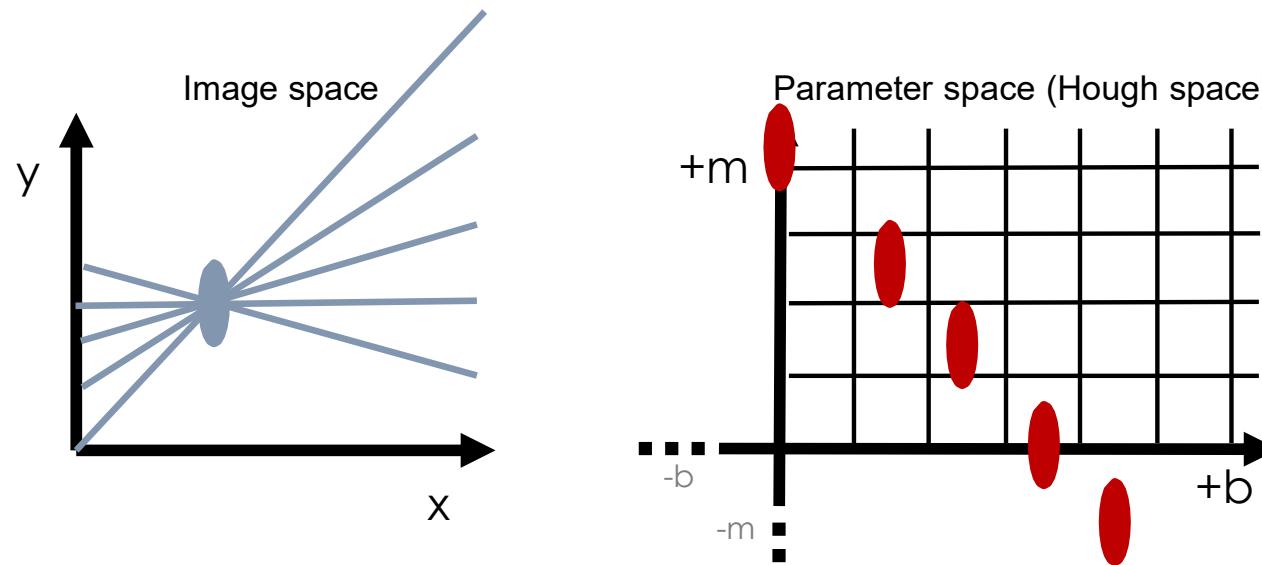
Hough Transform: Step back

- Hough space represents all possible lines.
- With gradient information constraint:
 - Edge is single point in Hough space.
- Without gradient orientation information?



Hough Transform: Step back

- Hough space represents all possible lines.
- With gradient information constraint:
 - Edge is single point in Hough space.
- Without gradient orientation information?



Hough Transform: Step back

- Hough space represents all possible lines.
- With gradient information constraint:
 - Edge is single point in Hough space.
- Without gradient orientation information?
 - Point is line in Hough space

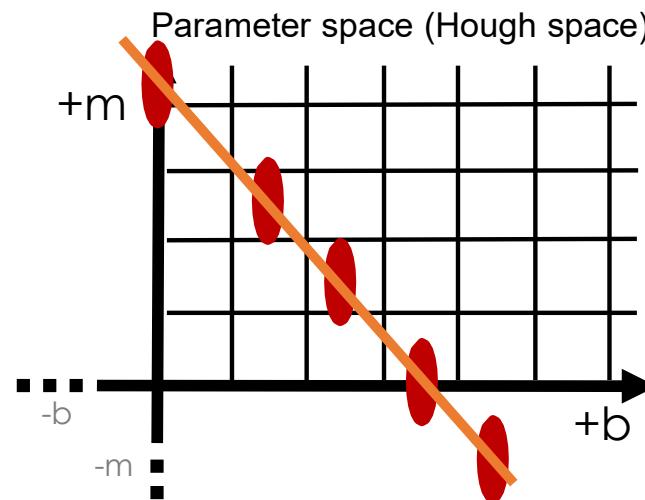
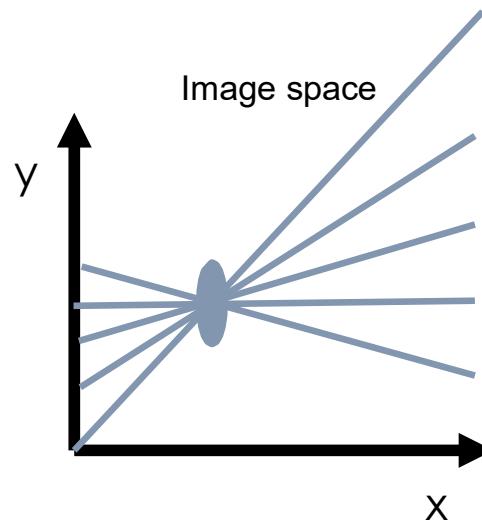
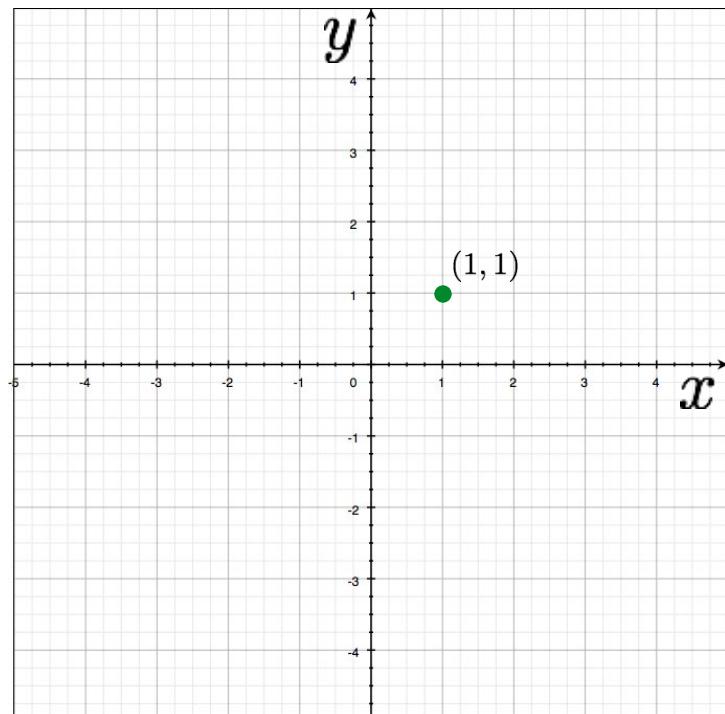


Image and parameter space

variables
 $y = mx + b$
parameters



a point
becomes a
line

variables
 $y - mx = b$
parameters

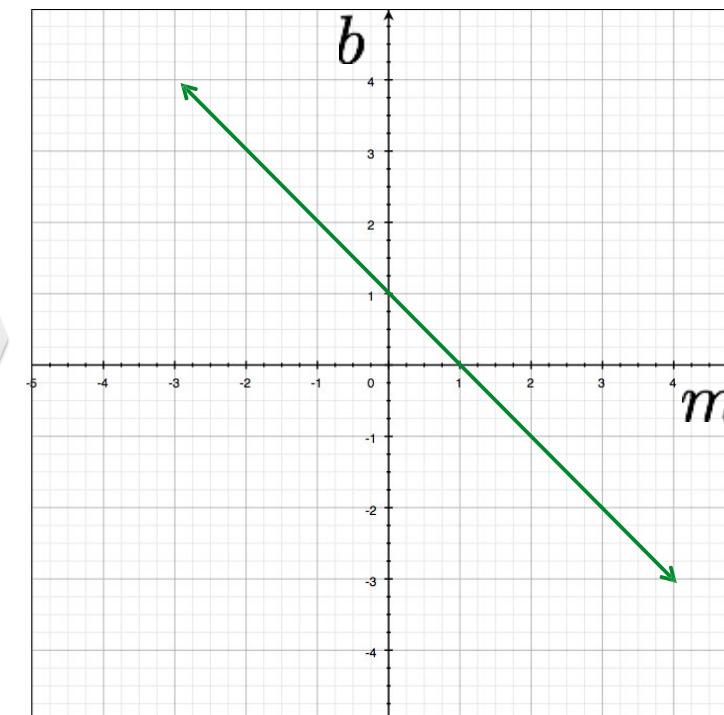
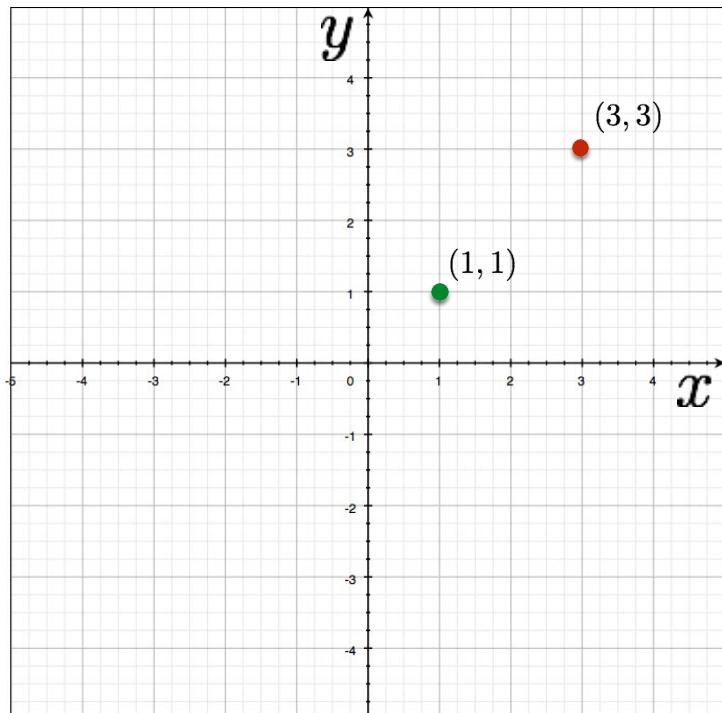


Image space

Parameter space

Image and parameter space

variables
 $y = mx + b$
parameters



two points
become
?

variables
 $y - mx = b$
parameters

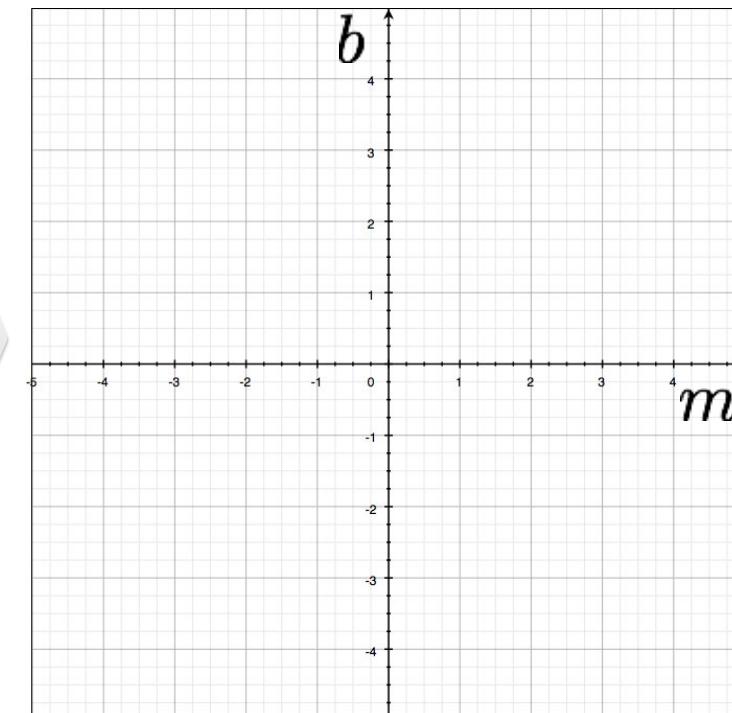
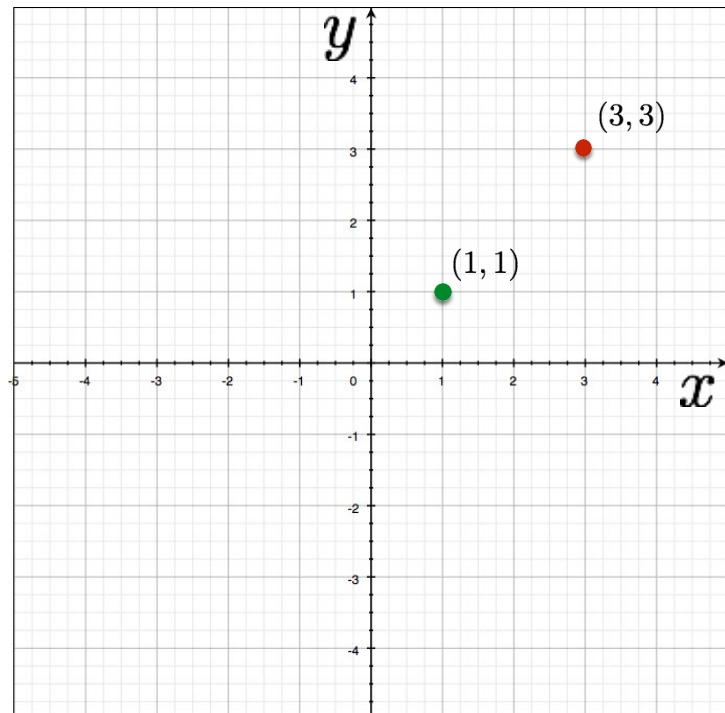


Image space

Parameter space

Image and parameter space

variables
 $y = mx + b$
parameters



two points
become
?

variables
 $y - mx = b$
parameters

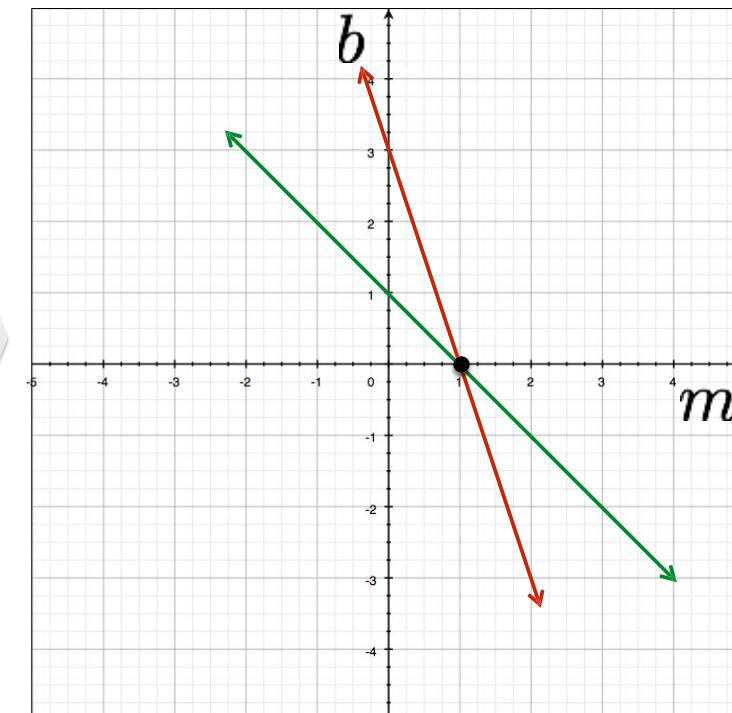
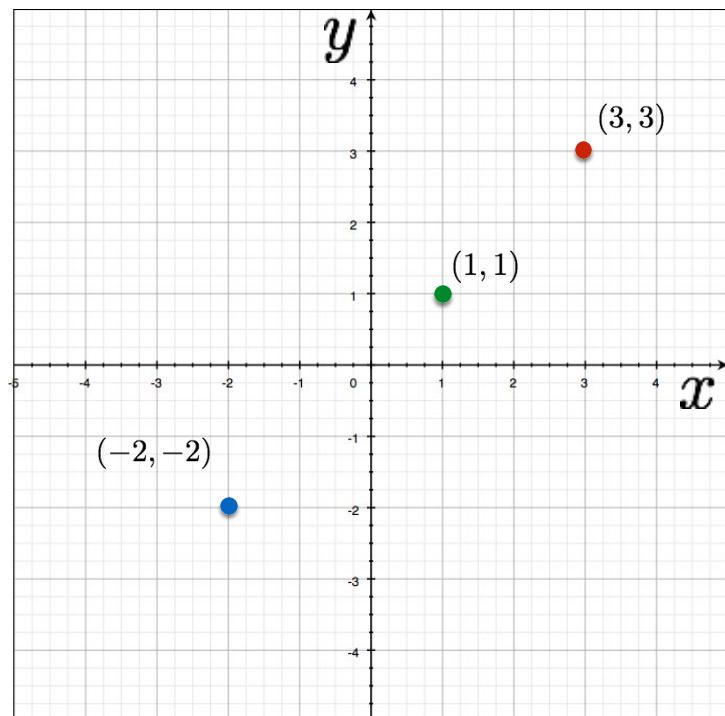


Image space

Parameter space

Image and parameter space

variables
 $y = mx + b$
parameters



three points
become
?

variables
 $y - mx = b$
parameters

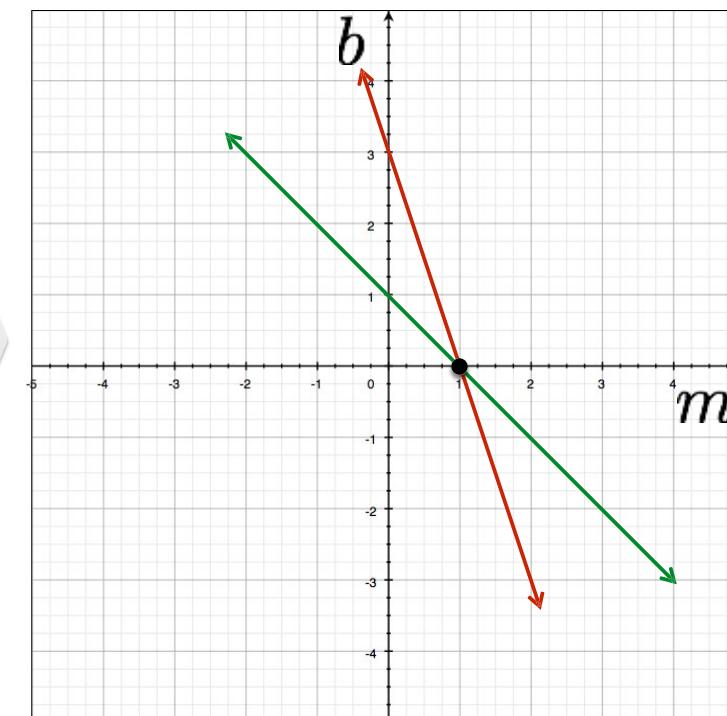
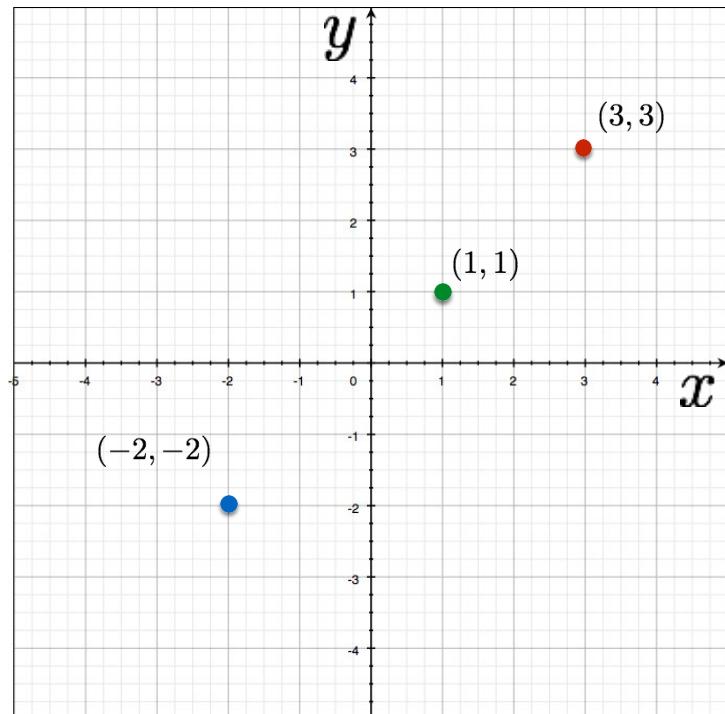


Image space

Parameter space

Image and parameter space

variables
 $y = mx + b$
parameters



three points
become
?

variables
 $y - mx = b$
parameters

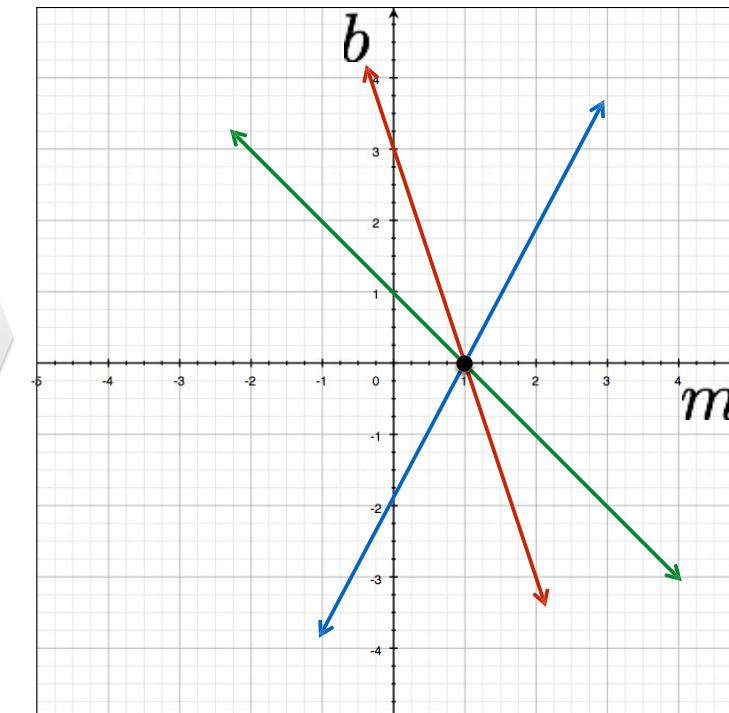


Image space

Parameter space

Image and parameter space

variables
 $y = mx + b$
parameters

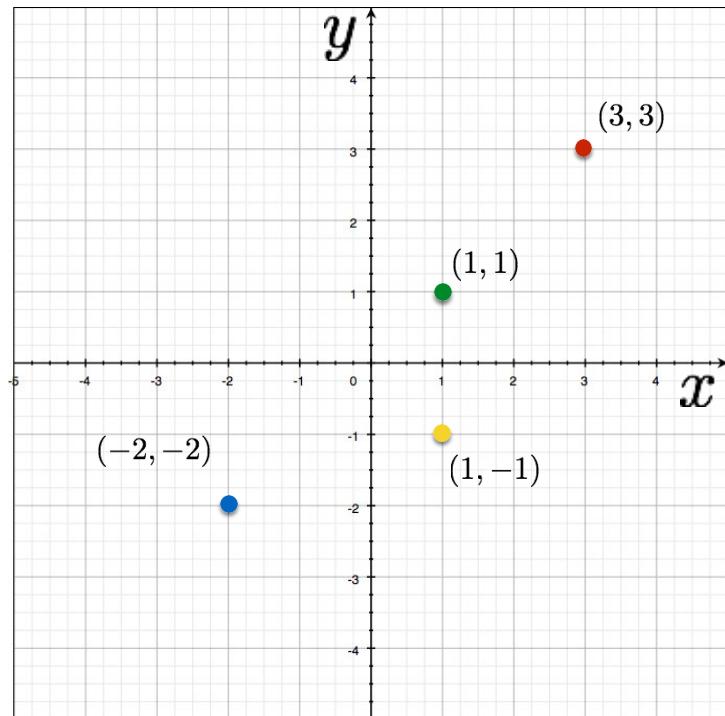
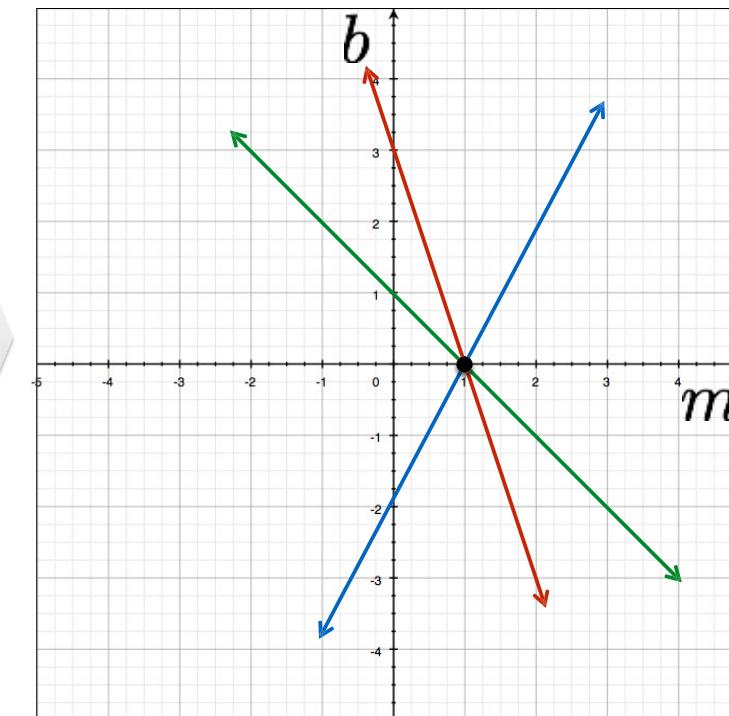


Image space

variables
 $y - mx = b$
parameters



Parameter space

four points
become
?

Image and parameter space

variables
 $y = mx + b$
parameters

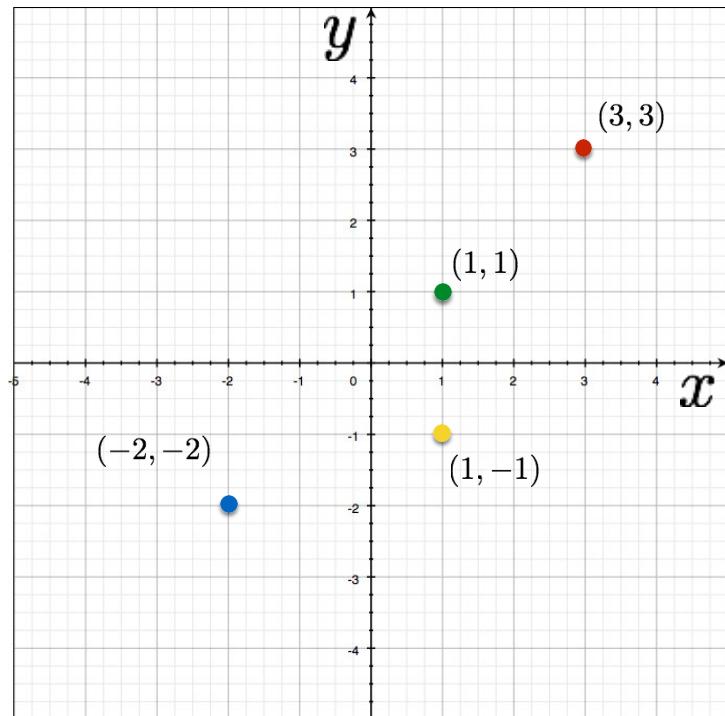
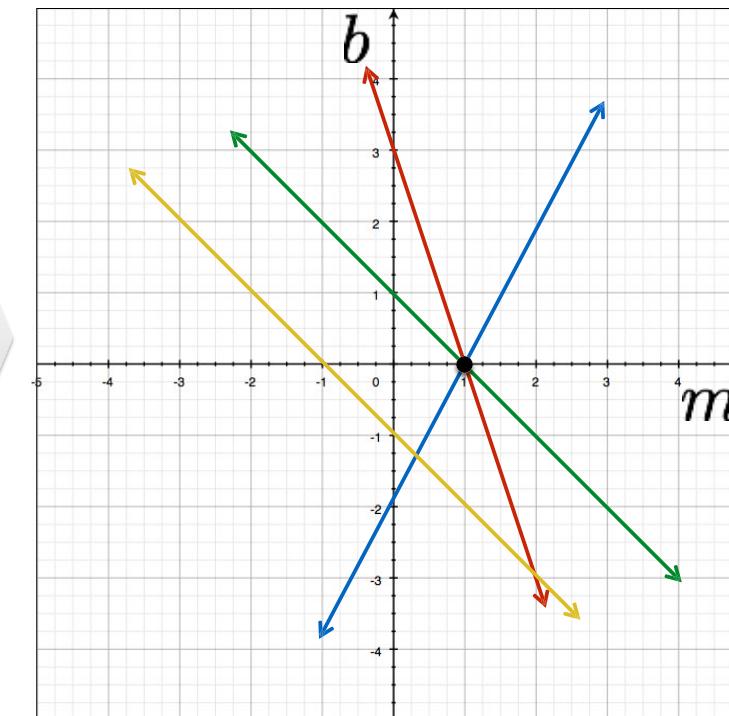


Image space

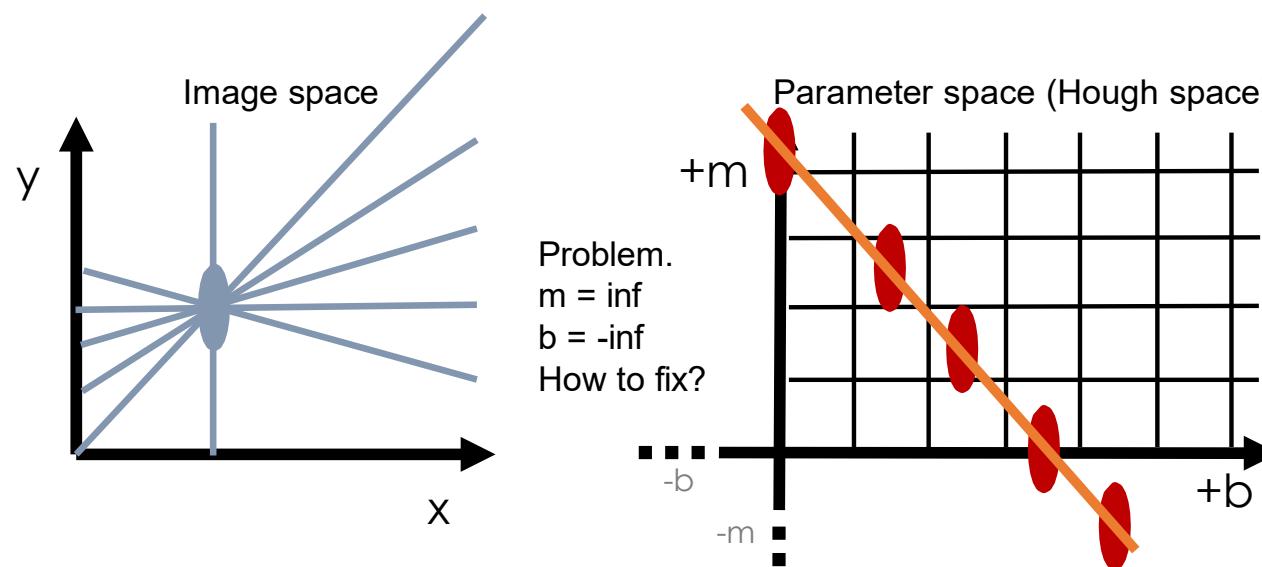
variables
 $y - mx = b$
parameters



Parameter space

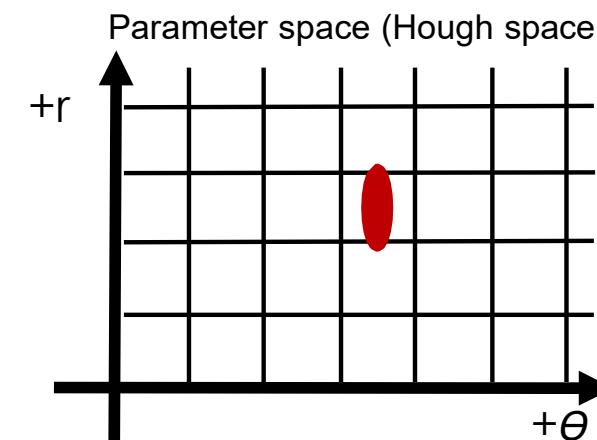
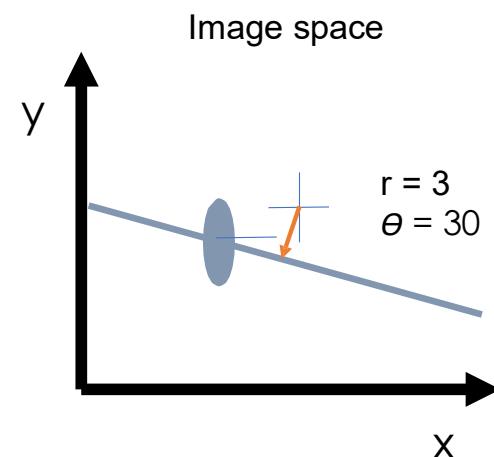
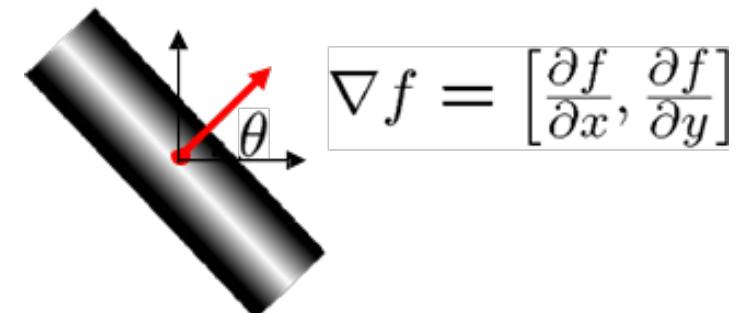
Hough Transform: Step back

- Hough space represents all possible lines
- With gradient information constraint:
 - Edge is single point in Hough space
- Without gradient orientation information?
 - How big is Hough space?



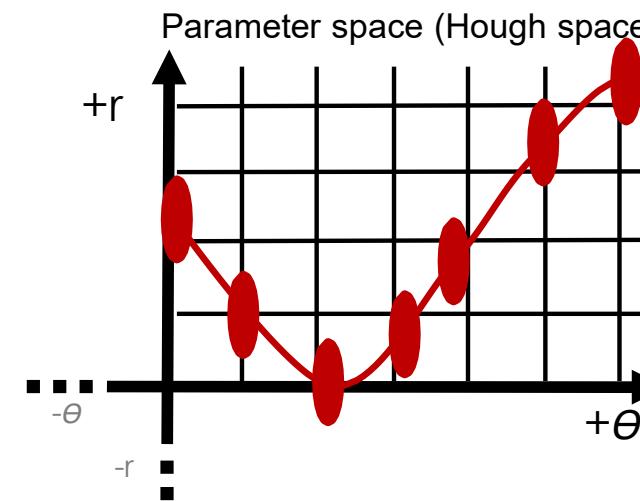
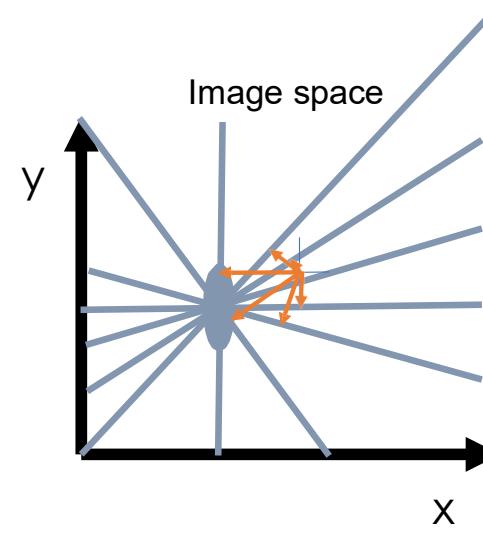
Hough Transform: Line Normal Form

- Use $\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$
 - Space is 0 to 360
- Use $r = \text{distance to line from some origin}$
 - $r_i = x_i \cos \theta_i + y_i \sin \theta_i$
 - Space is $\pm \sqrt{\max_x^2 + \max_y^2}$



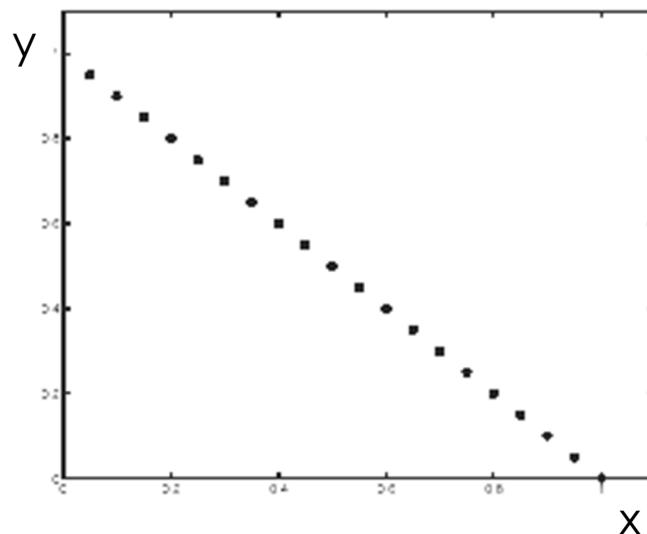
Hough Transform: Line Normal Form

- In this line form, unoriented edge draws a sinusoid in Hough space:

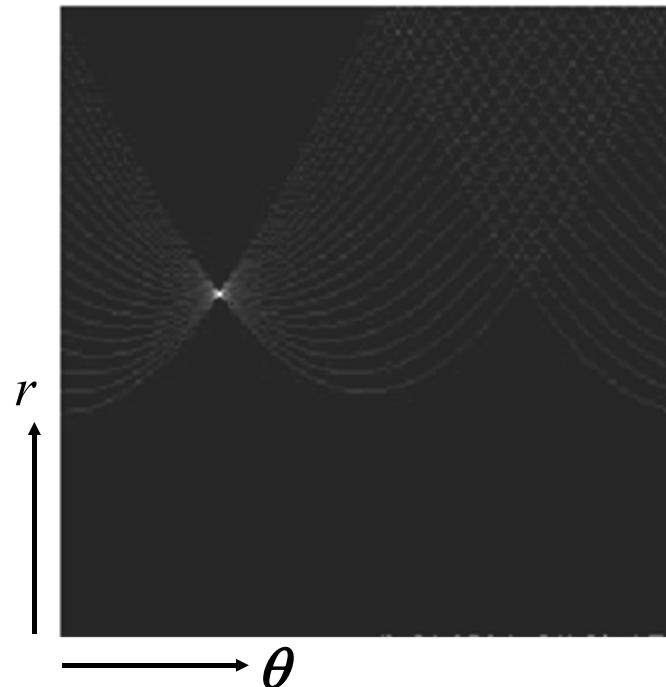


Hough transform - experiment

Next few images *ignore* edge orientation.
Each point is one sinusoid.

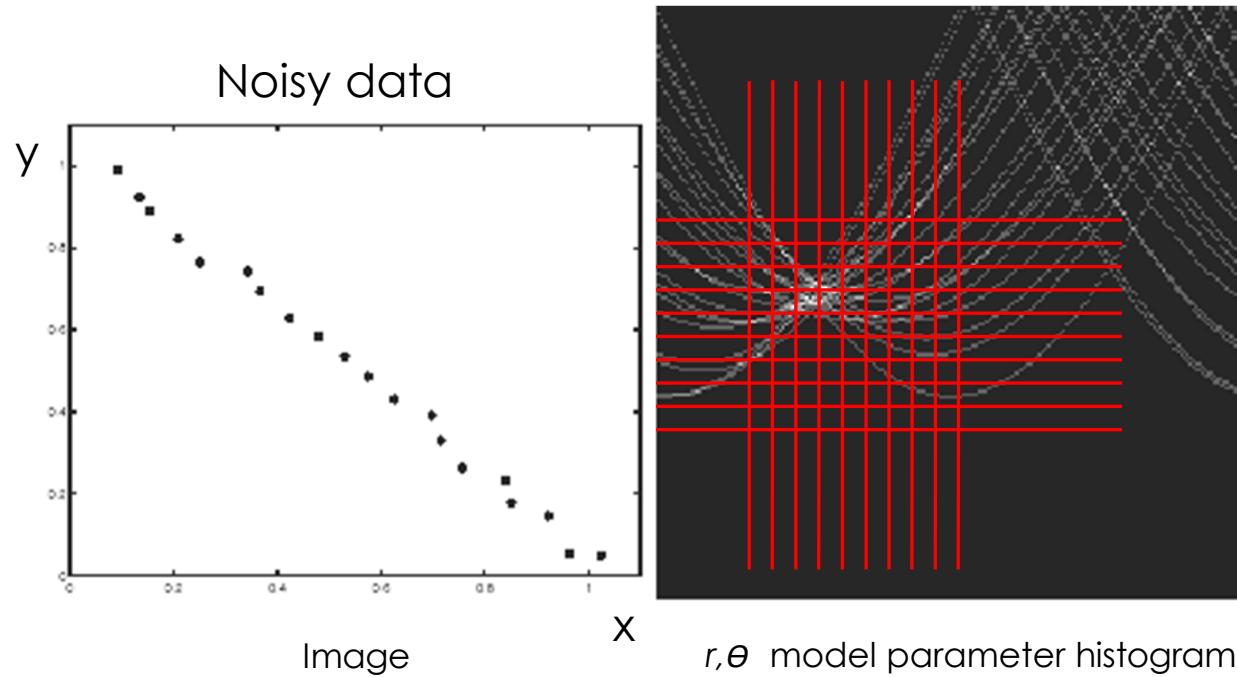


Image



r, θ model parameter histogram

Hough transform - experiment



Need to adjust grid size or smooth

- Practical considerations
 - Bin size
 - Smoothing
 - Finding multiple lines
 - Finding line segments

Hough transform example

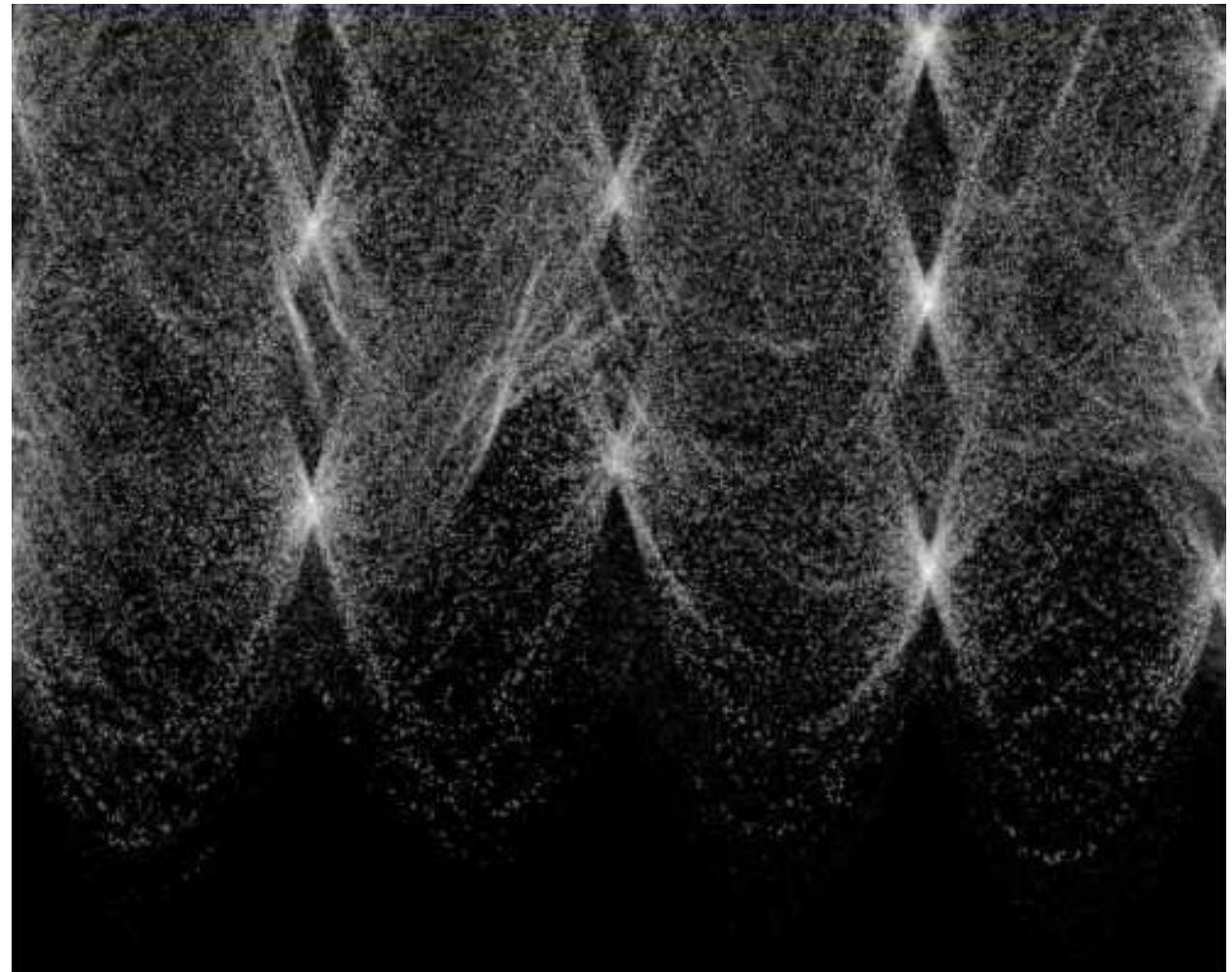
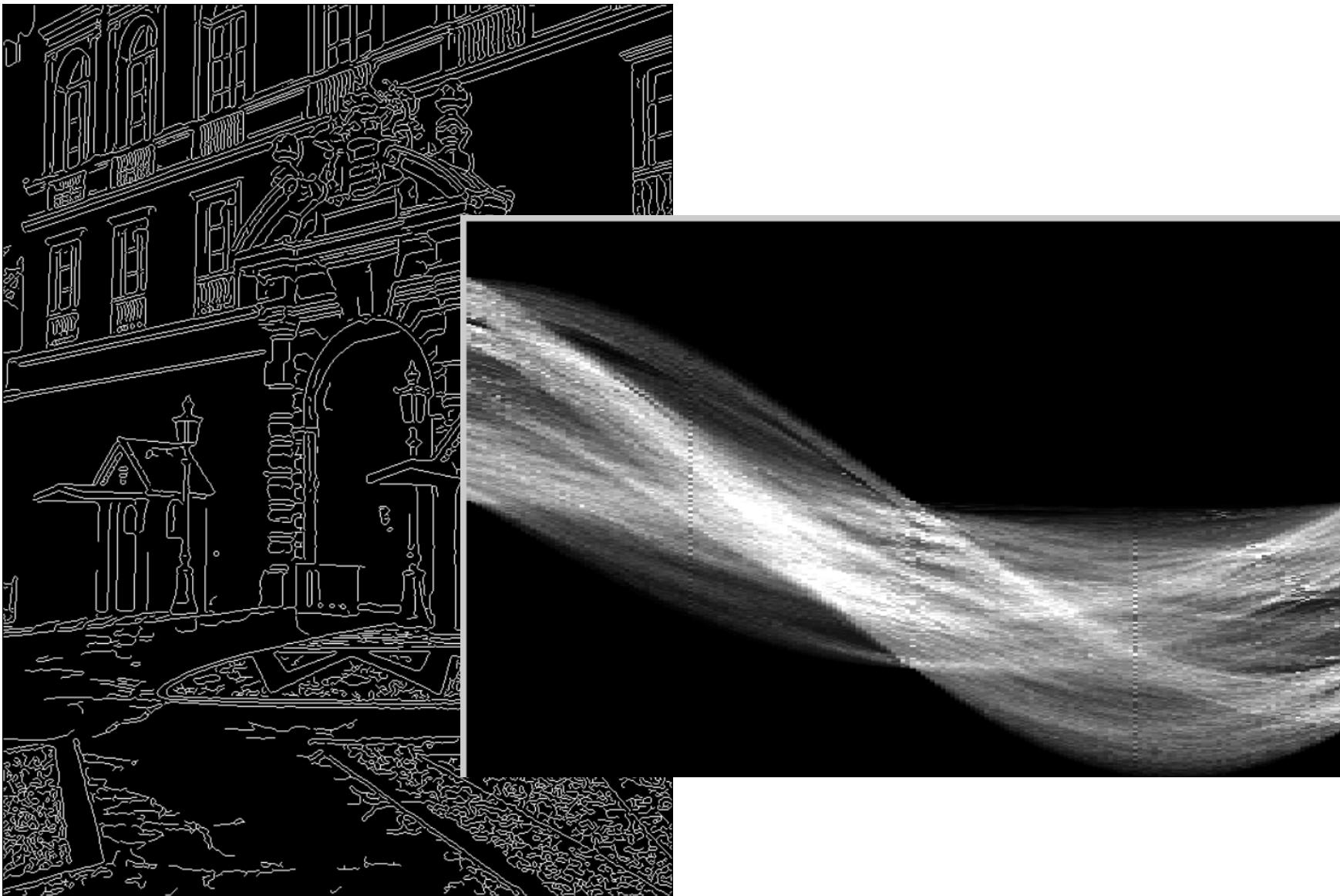


Image → Canny

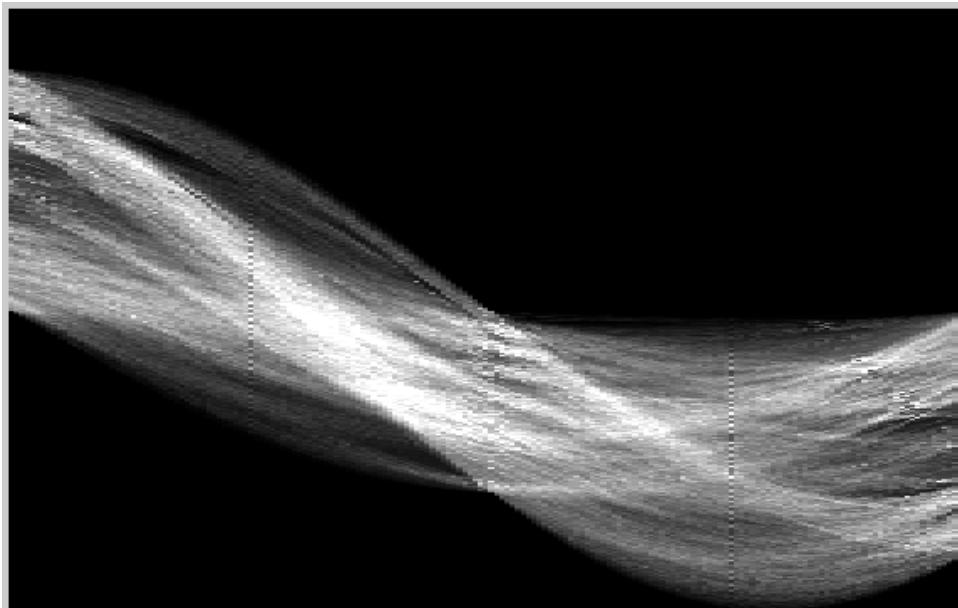


Canny → Hough votes

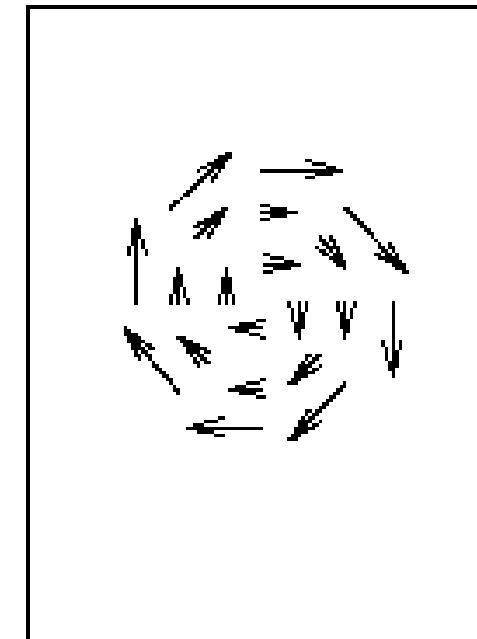
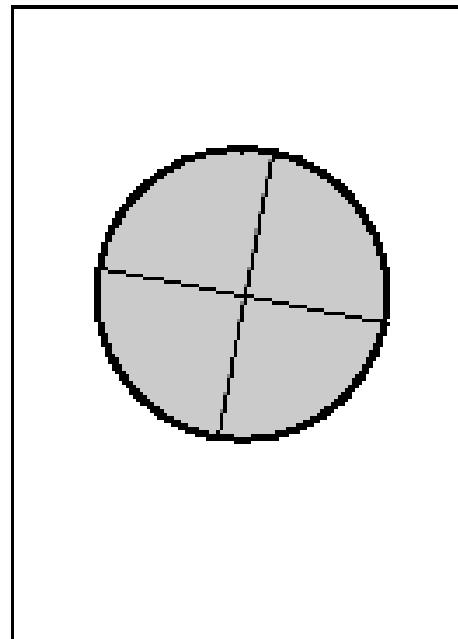
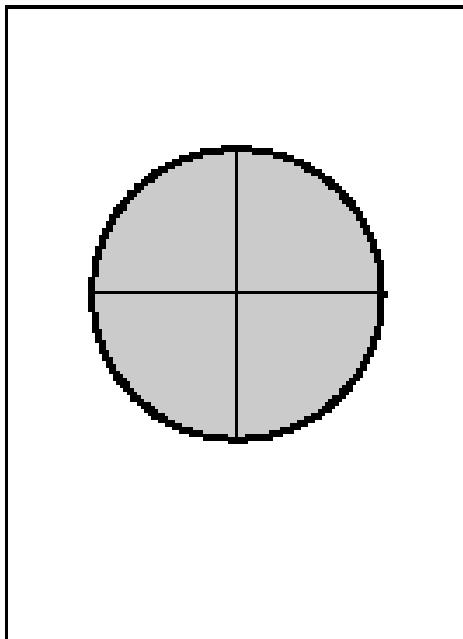


Hough votes → Edges

Find peaks and post-process.



Optical Flow



Fundamental Equations of Computer Vision

I. Image Filtering

$$h[m, n] = \sum_{k, l} f[k, l] I[m + k, n + l]$$

2. Optical Flow

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

3. Camera Geometry

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \ \mathbf{t}] \mathbf{X} \quad x^T F x' = 0$$

4. Machine Learning

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2. \quad y = \varphi(\sum_{i=1}^n w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

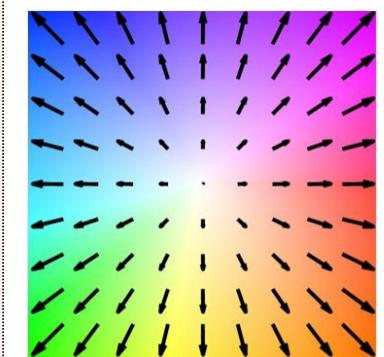
Optical Flow: Dense Correspondence Over Time



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



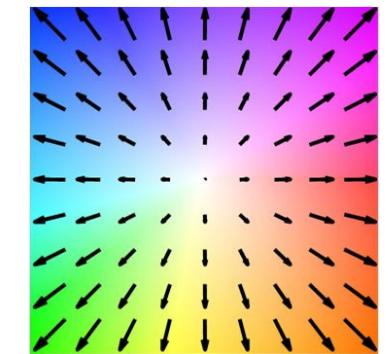
Optical flow (2D motion vector)



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



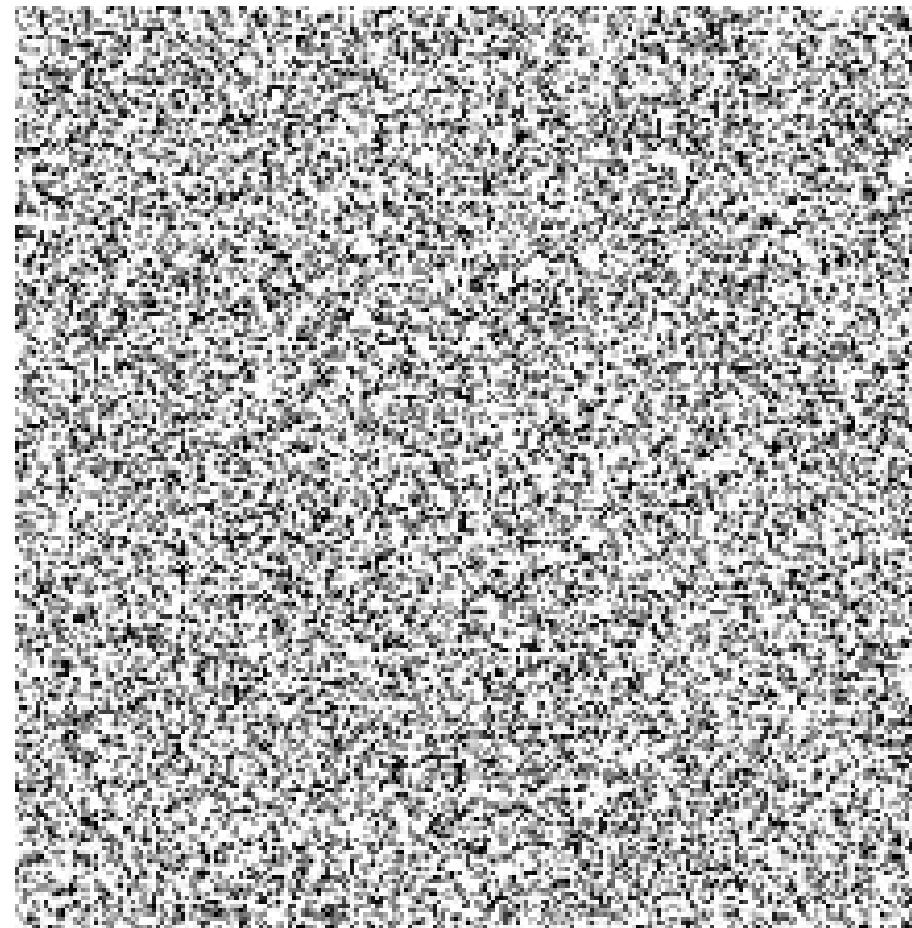
https://www.youtube.com/watch?v=e_TeY6QRp4c



Color key

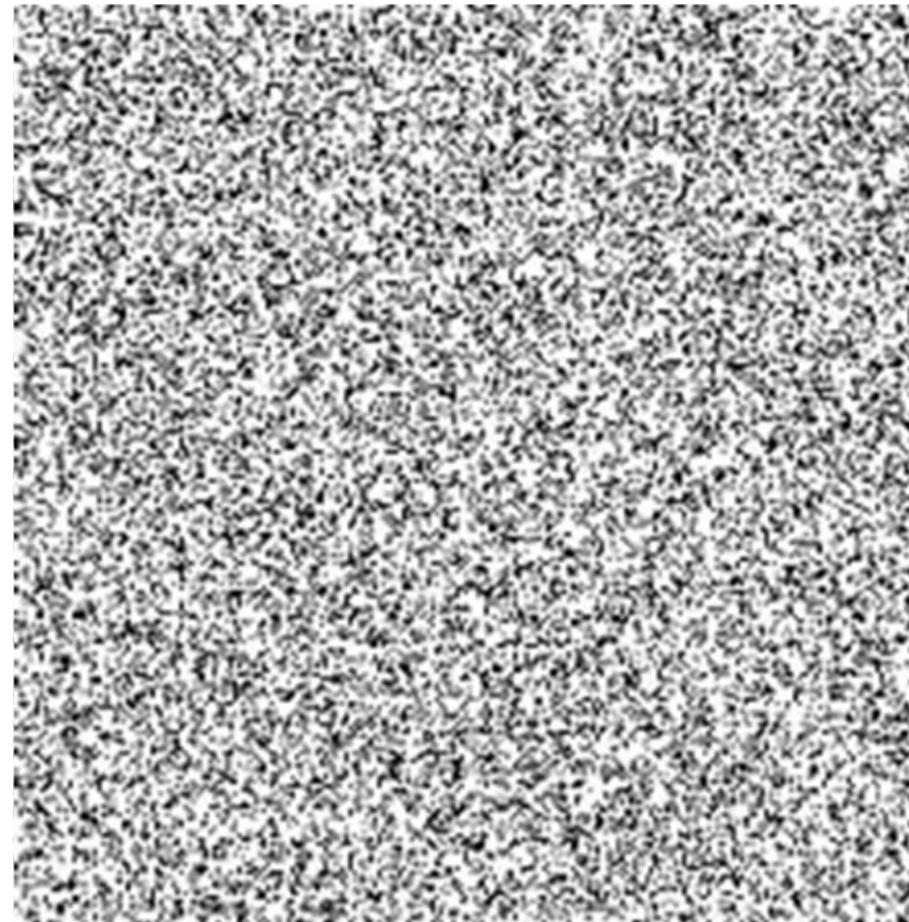
Correspondence and perceptual organization

Sometimes correspondence/motion is the only cue



Correspondence and perceptual organization

Sometimes correspondence/motion is the only cue





<https://www.youtube.com/watch?v=mNrqvcS0oI0>

Applications of optical flow



Applications of optical flow



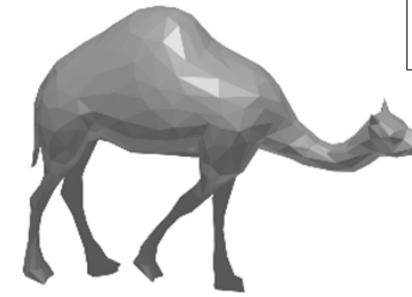
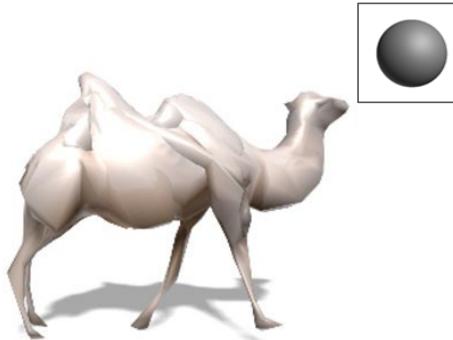
<https://youtu.be/MjViy6kyiqs?si=zXenEX6O6VJkEaQF>

Super SloMo [Jiang, Sun, et al. CVPR 2018 Spotlight] Incorporated into **NVIDIA NGX** SDK for the Turing GPU.

Face Unblur for Pixel 6



Articulated shape reconstruction from a monocular video



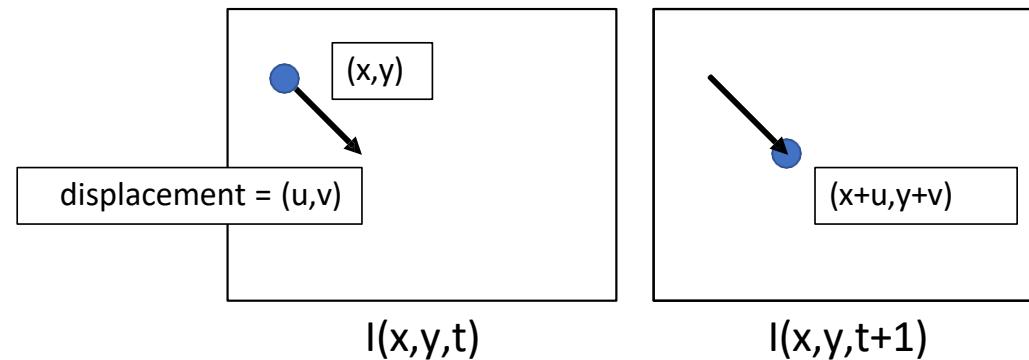
A-CSM: Kulkarni et al. CVPR 2020



VIBE: Kocabas et al. CVPR 2020

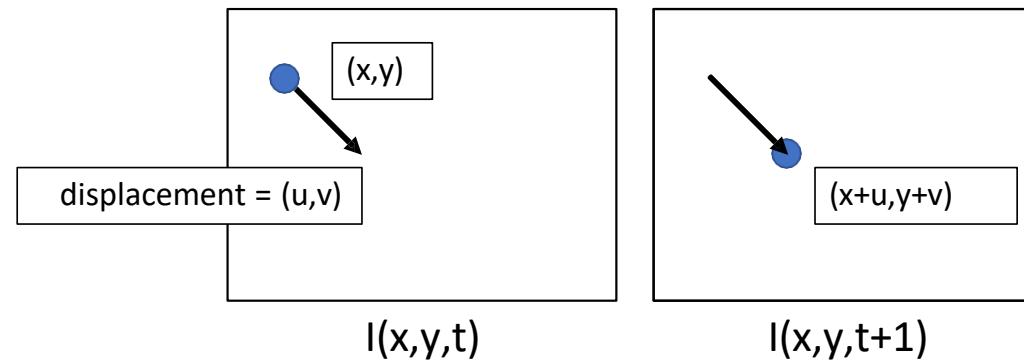
Challenge: Solving non-rigid 3D shape from 2D measurements without template or category prior is highly *under-constrained*

Optical Flow



Brightness constancy: $I(x, y, t) = I(x + u, y + v, t + 1)$

Optical Flow



Brightness constancy: $I(x, y, t) = I(x + u, y + v, t + 1)$

Recall Taylor Expansion: $I(x + u, y + v, t + 1) = I(x, y, t) + I_x u + I_y v + I_t \dots$

Optical Flow Equation

$$\begin{aligned} I(x + u, y + v, t + 1) &= I(x, y, t) \\ 0 &= I(x + u, y + v, t + 1) - I(x, y, t) \quad \text{Taylor} \\ &\approx I(x, y, t) + I_t + I_x u + I_y v - I(x, y, t) \quad \text{Expansion} \\ &= I_t + I_x u + I_y v \\ &= I_t + \nabla I \cdot [u, v] \end{aligned}$$

When is this approximation **bad?**
u or v big

Optical Flow Equation

Brightness constancy equation

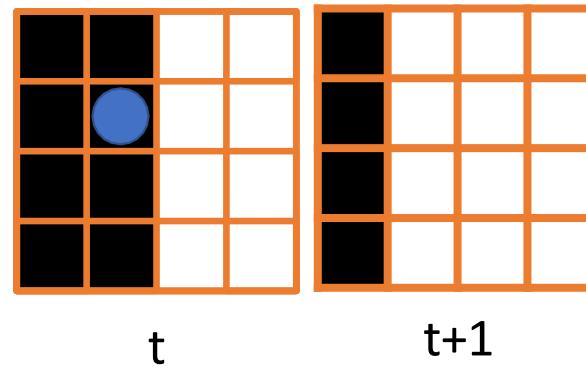
$$I_x u + I_y v + I_t = 0$$

What do static image gradients have
to do with motion estimation?



Brightness Constancy Example

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



$$I_t = 1 - 0 = 1$$

@ $I_y = 0$

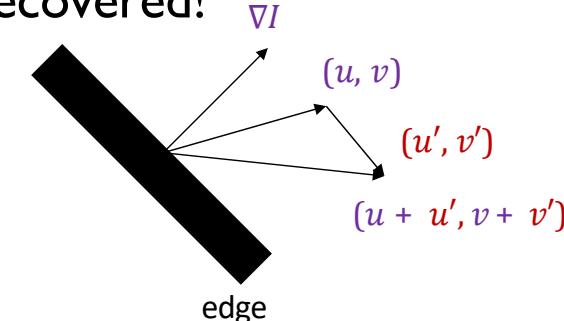
$$I_x = 1 - 0 = 1$$

What's u ? What's v ?

The Brightness Constancy Constraint

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

- Given the gradients I_x , I_y and I_t , can we uniquely recover the motion (u, v) ?
 - One equation, two unknowns
 - Suppose (u, v) satisfies the constraint: $\nabla I \cdot (u, v) + I_t = 0$
 - Then $\nabla I \cdot (u + u', v + v') + I_t = 0$ for any (u', v') s.t. $\nabla I \cdot (u', v') = 0$
 - Interpretation: the component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be recovered!



Solving Ambiguity – Lucas Kanade

- 2 unknowns $[u, v]$, 1 eqn per pixel How do we get more equations?
- Assume **spatial coherence**: pixel's neighbors have move together / have same $[u, v]$
- 5x5 window gives 25 new equations

$$I_t + I_x u + I_y v = 0$$
$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Solving for u, v

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$
$$\underset{25 \times 2}{A} \underset{2 \times 1}{d} = \underset{25 \times 1}{b}$$

What's the solution?

$$(A^T A) d = A^T b \rightarrow d = (A^T A)^{-1} A^T b$$

Intuitively, need to solve (sum over pixels in window)

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$\underset{A^T A}{A^T A} \underset{A^T b}{d} = \underset{b}{b}$$

Challenges for traditional methods



Large motion, motion blur



Textureless regions

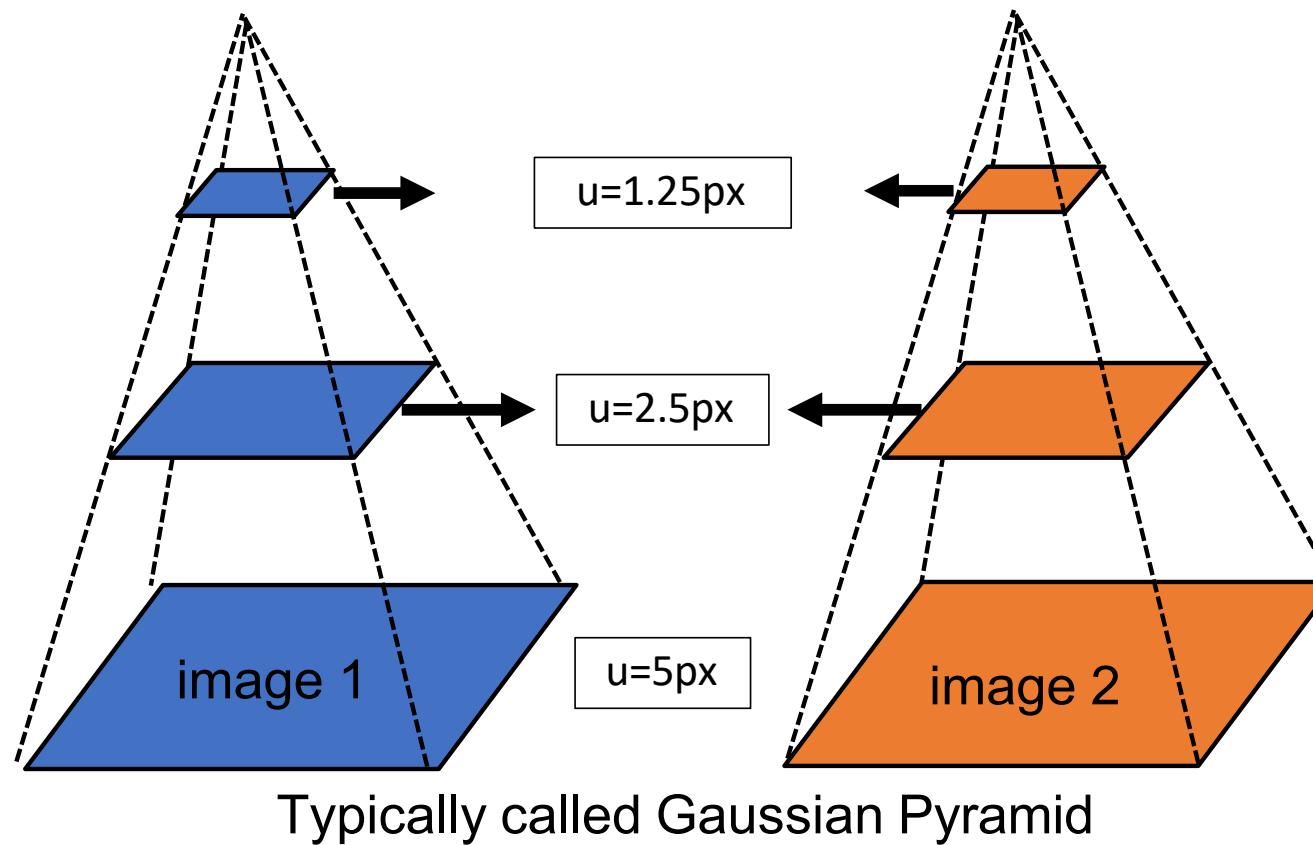


Occlusions

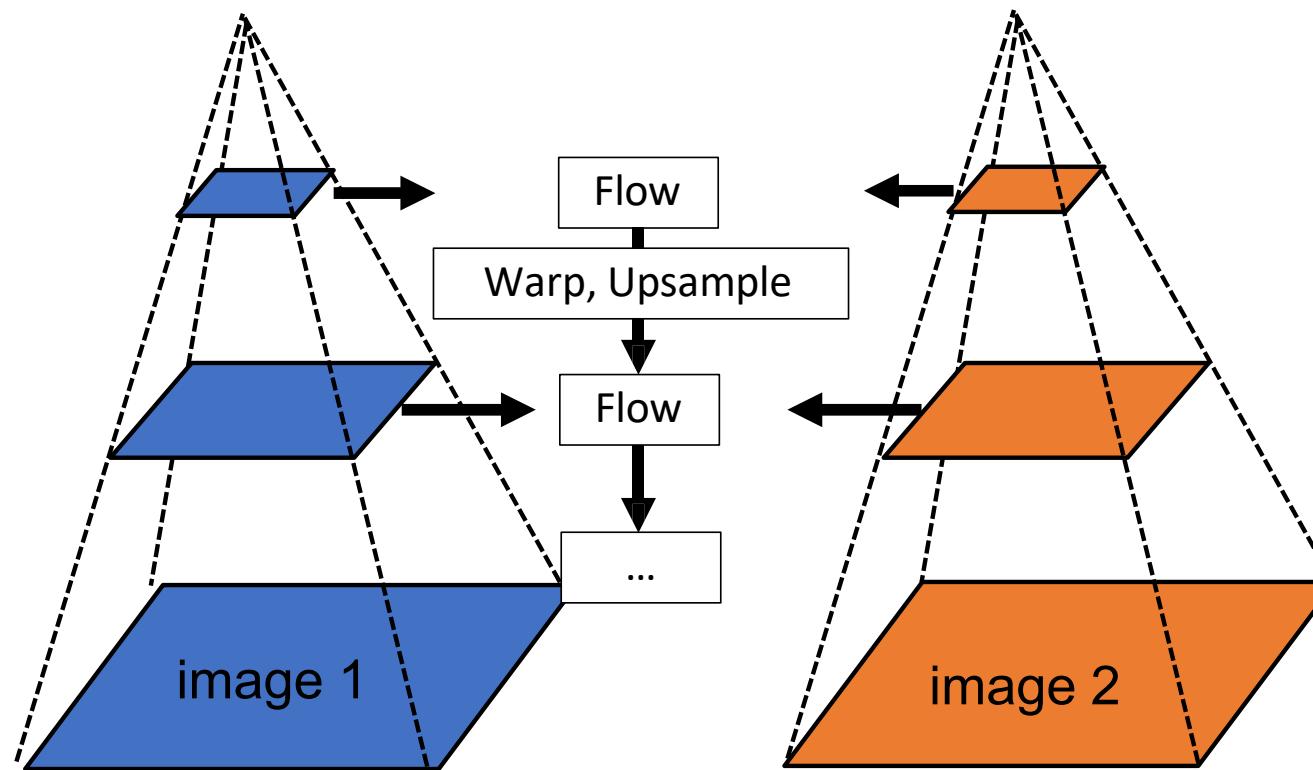


Lighting changes, noise ...

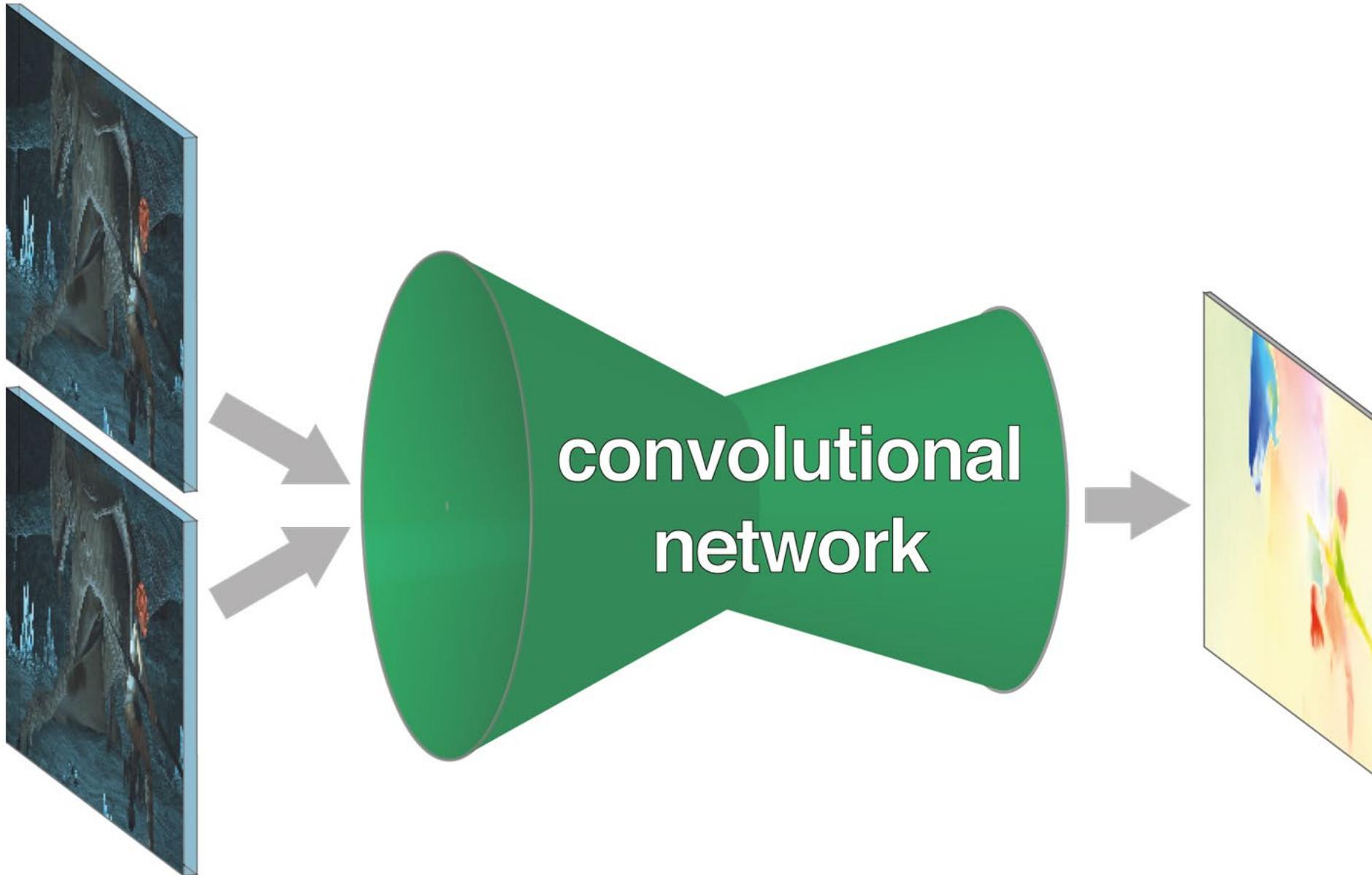
Coarse-to-fine optical flow estimation



Coarse-to-fine optical flow estimation



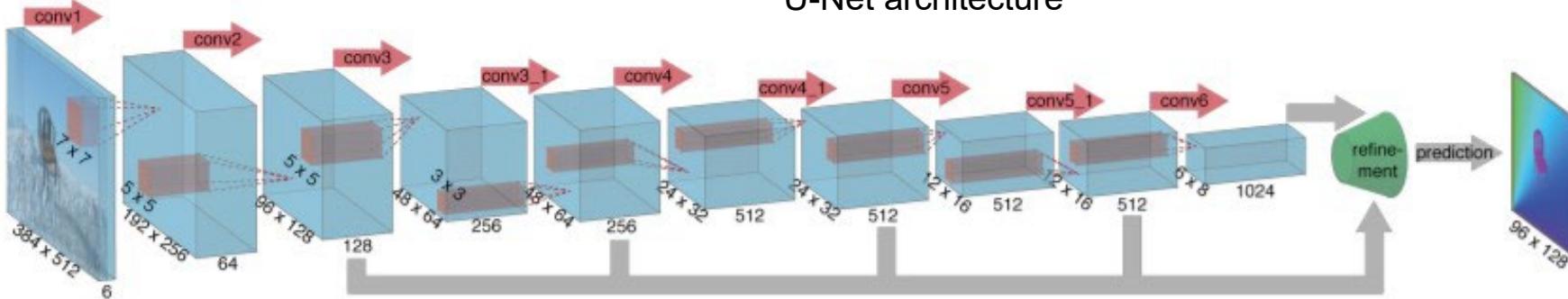
Typical DL Pipeline for Optical Flow



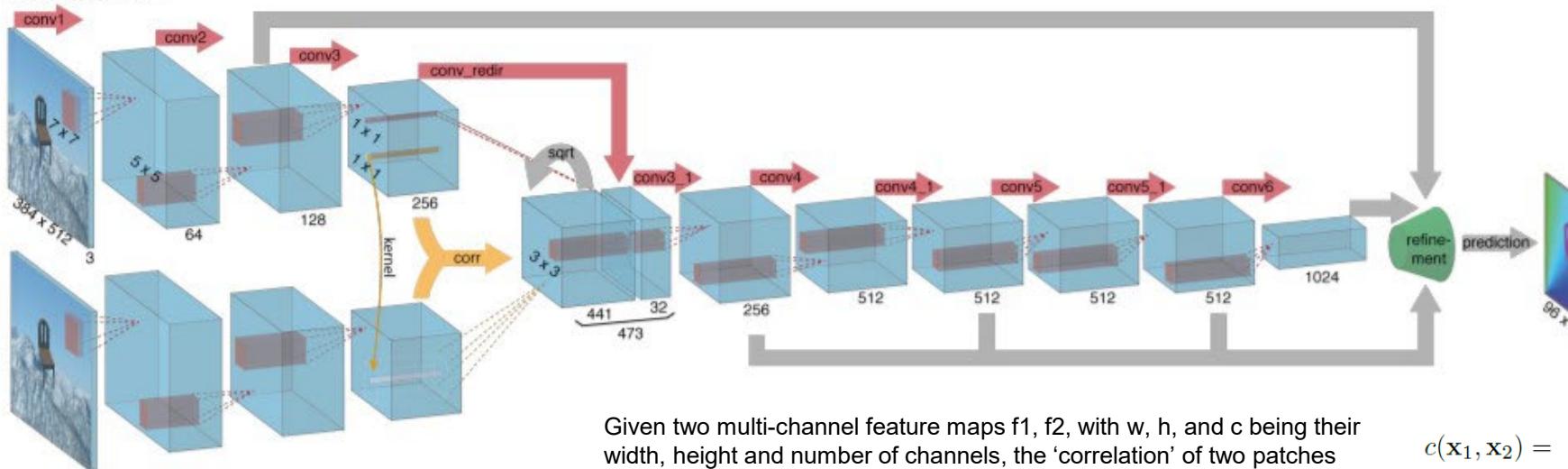
FlowNetS: mapping from images to flow

[Dosovitskiy *et al.* ICCV'15]

FlowNetSimple



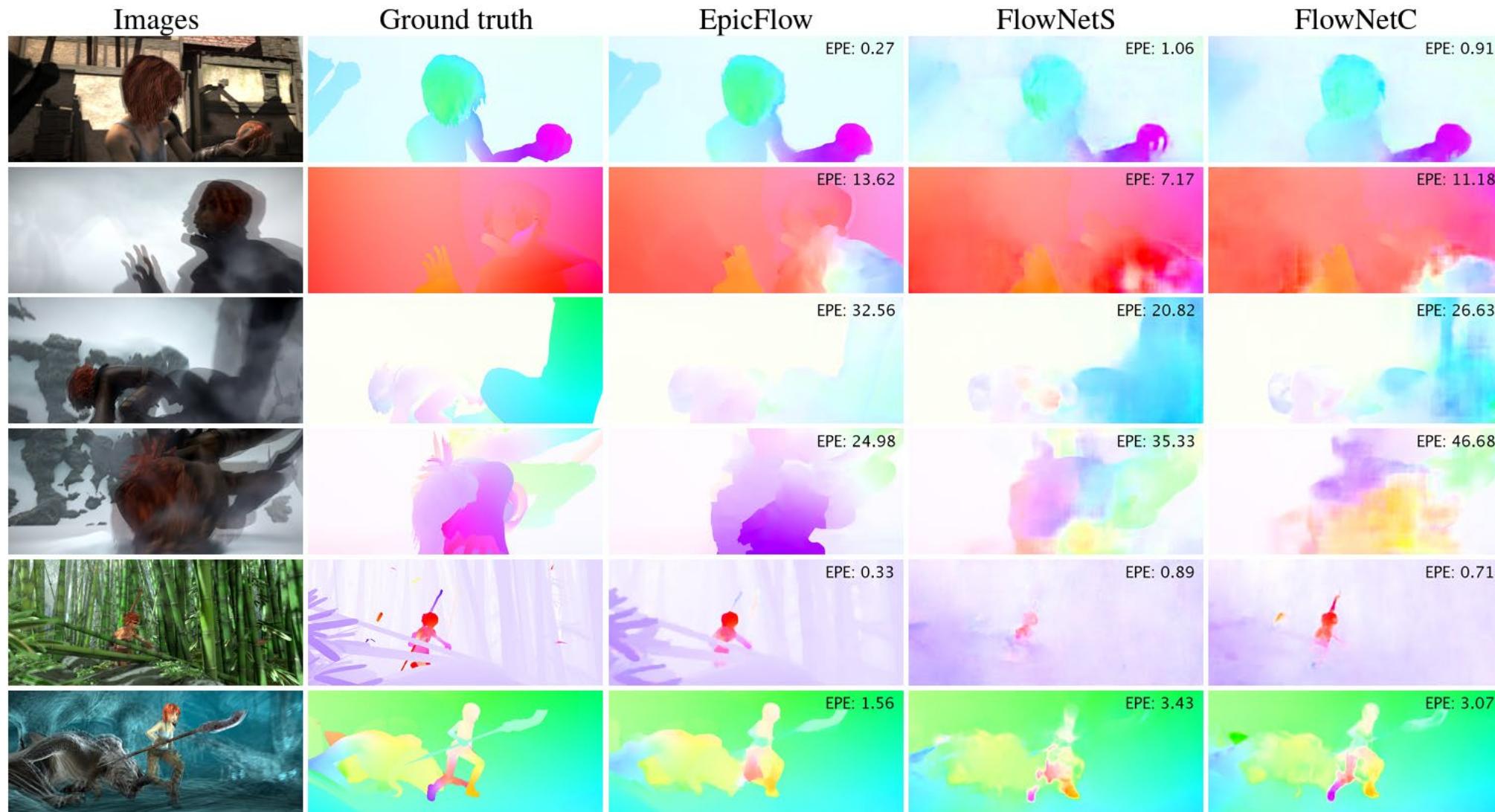
FlowNetCorr



Given two multi-channel feature maps f_1, f_2 , with w, h , and c being their width, height and number of channels, the 'correlation' of two patches centered at x_1 in the first map and x_2 in the second map is then defined as:

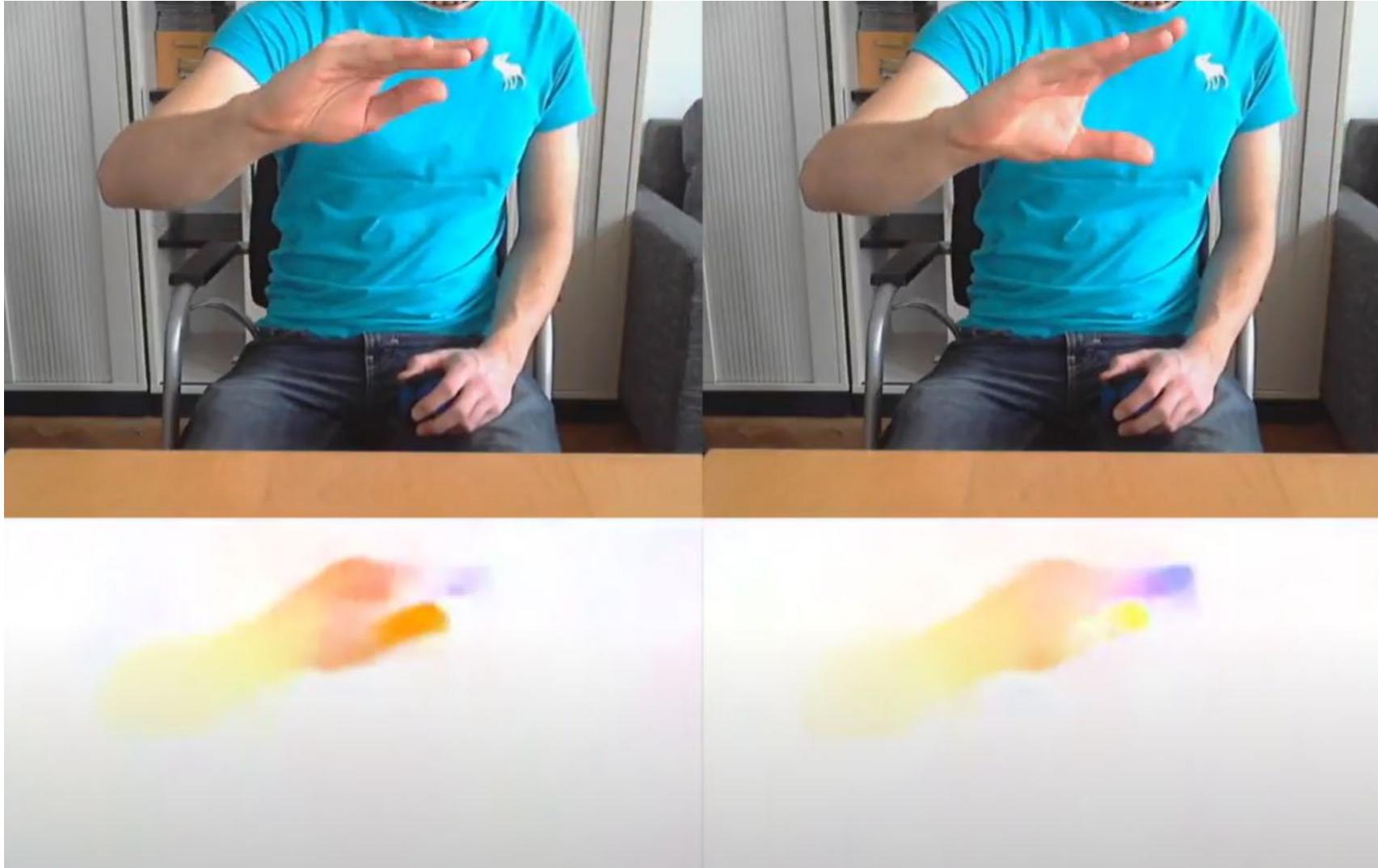
$$c(x_1, x_2) = \sum_{o \in [-k, k] \times [-k, k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle$$

FlowNetS: mapping from images to flow



EPE: endpoint error

FlowNetS: mapping from images to flow



https://youtu.be/k_wkDLJ8lJE?si=wJj1RePbw-NA75lU