CIS 5800

Machine Perception

Instructor: Lingjie Liu

Lec 17: April 7, 2025

Robot Image Credit: Viktoriya Sukhanova © 123RF.com

Perspective painting masterpieces on the sidewalk

This is just a perspective image drawn without the “canvas” being approximately perpendicular to the artist’s line of sight.

Tracy Lee Stum

Perspective painting masterpieces on the sidewalk

Eduarda Rolero

Perspective painting masterpieces on the sidewalk

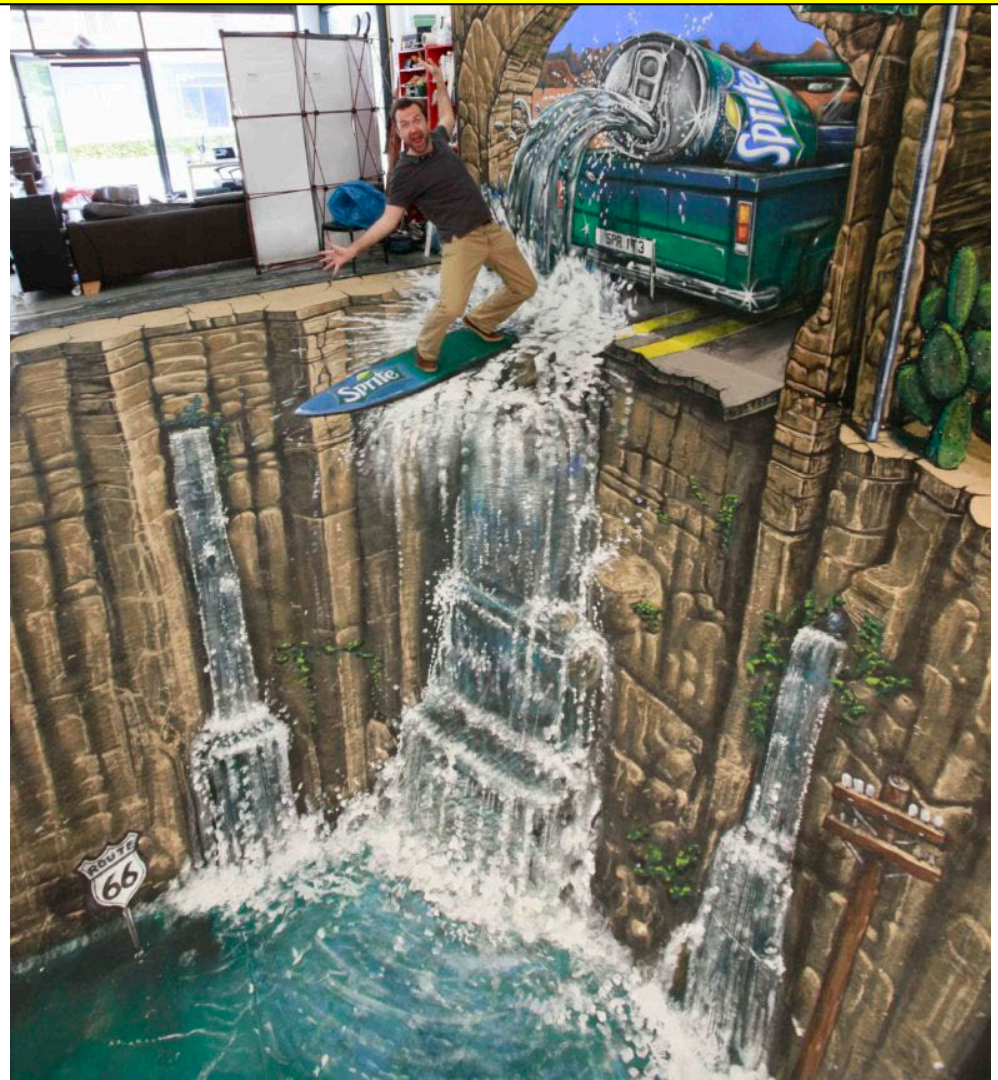
Kurt Wenner

Perspective painting masterpieces on the sidewalk

Edgar Mulle

Perspective painting masterpieces on the sidewalk

Depth from single image is all educated guesswork, based on perspective.
With some expert coaxing, the brain can be coaxed to hallucinate depth!



Joe Hill

Perspective painting masterpieces on the sidewalk

Depth from single image is all educated guesswork, based on perspective.
With some expert coaxing, the brain can be coaxed to hallucinate depth!

Here the canvas is not even a plane!

3D Anamorphic Street Art and a Video

But an alternative viewpoint can break the illusion!

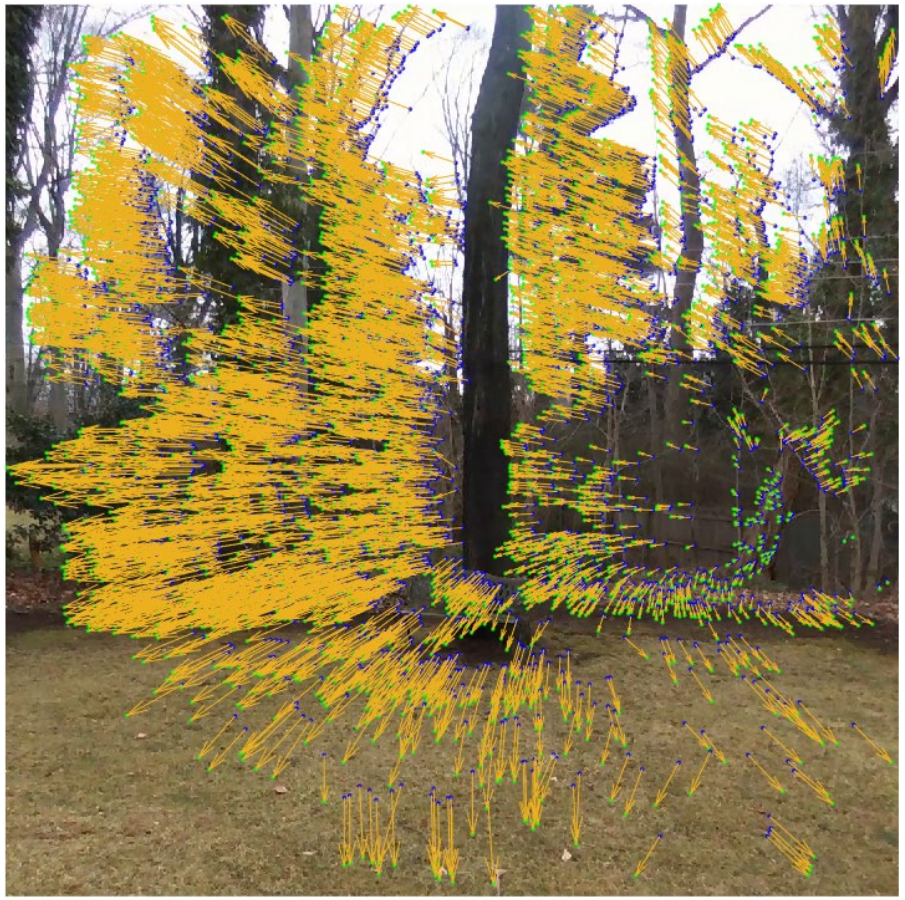
Perspective “Anamorphic art”: art that requires a special viewpoint for viewing

Leon Keer. https://www.youtube.com/watch?v=mxgBahscj2M

Recap: Optical Flow Correspondences

For some subset of patches in a scene, we want to compute how they move.

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



Recap: 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

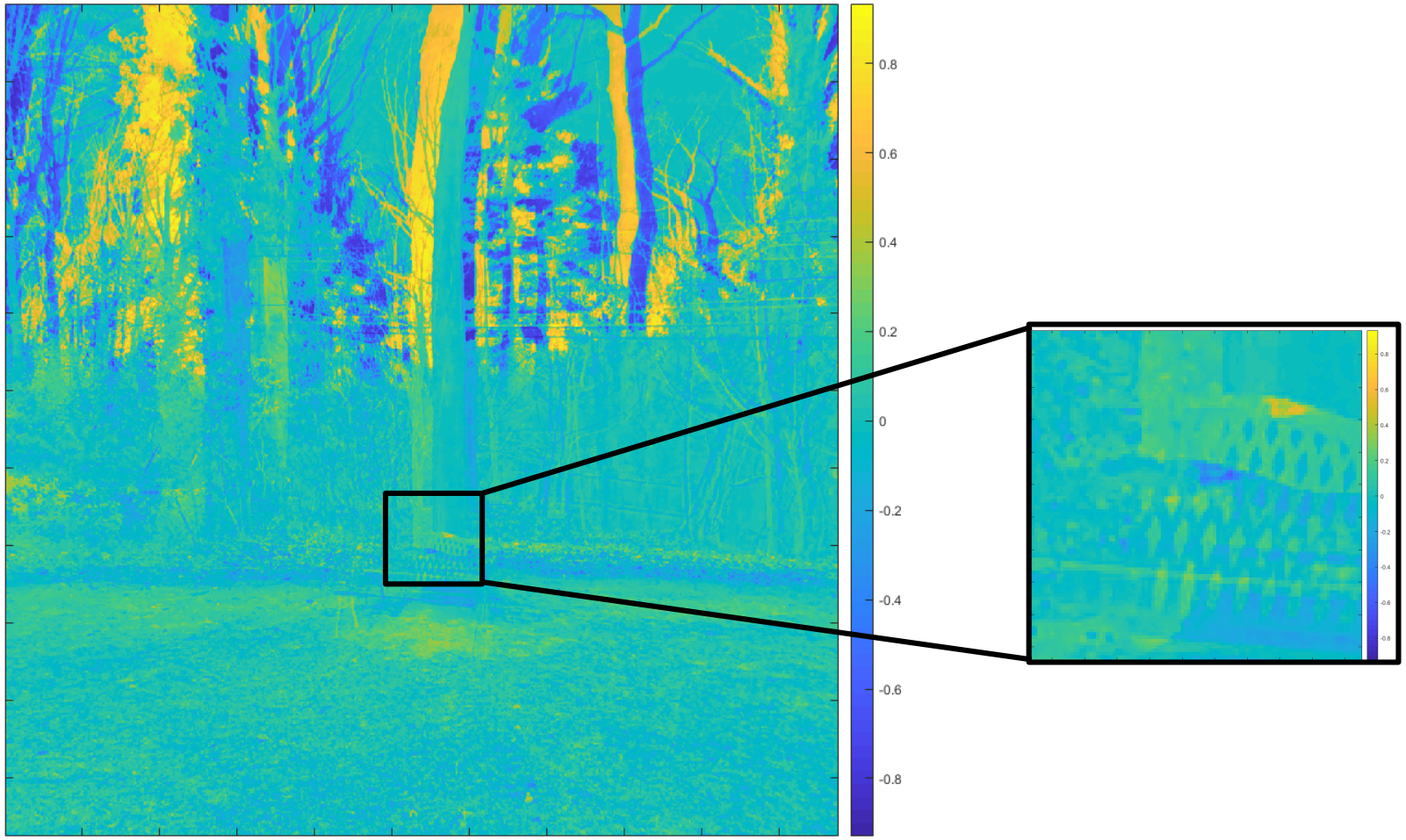
We want to select $\delta x, \delta y$ to minimize the sum of the square of this quantity over a neighborhood.

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

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

Recap: Term 1: Difference Image

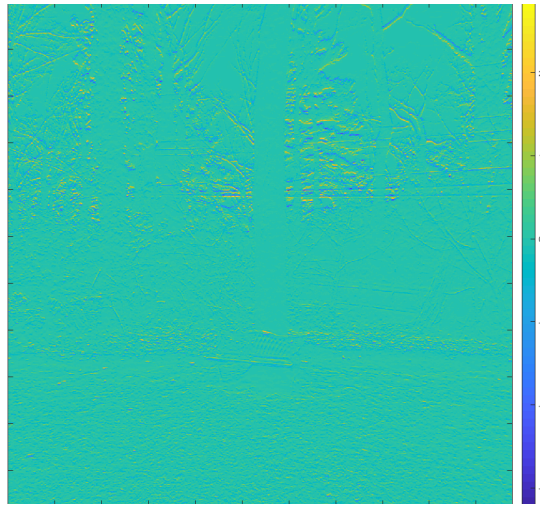
$$= \boxed{\Delta I_t(x,y)} - \nabla I_{t+1}(x,y)^T \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}$$



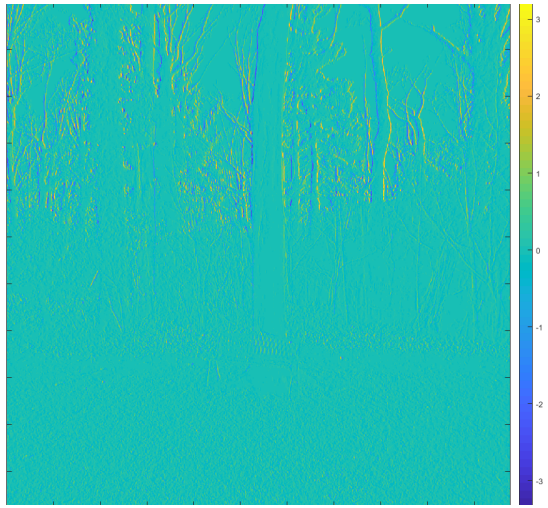
Recap: Term 2: Spatial Gradient

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

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

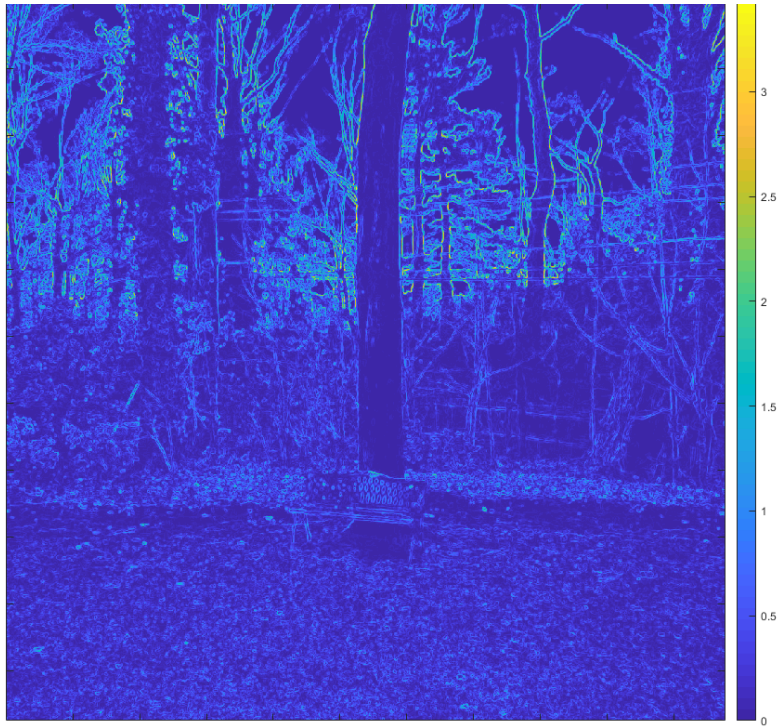


$\nabla_x I_{t+1}(x,y)$



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

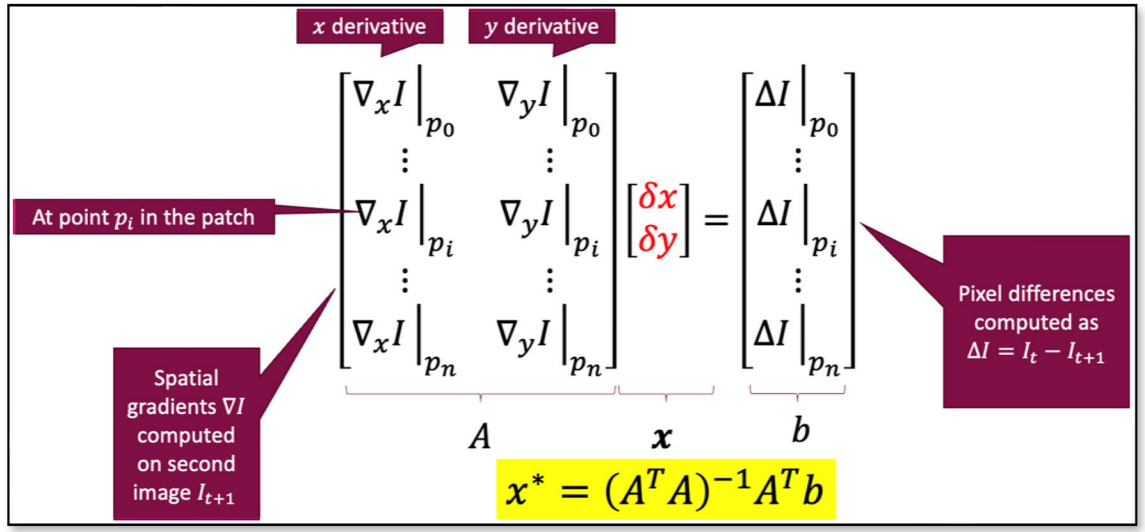
$\|\nabla I_{t+1}(x,y)\|$



Recap: Optical Flow Correspondences

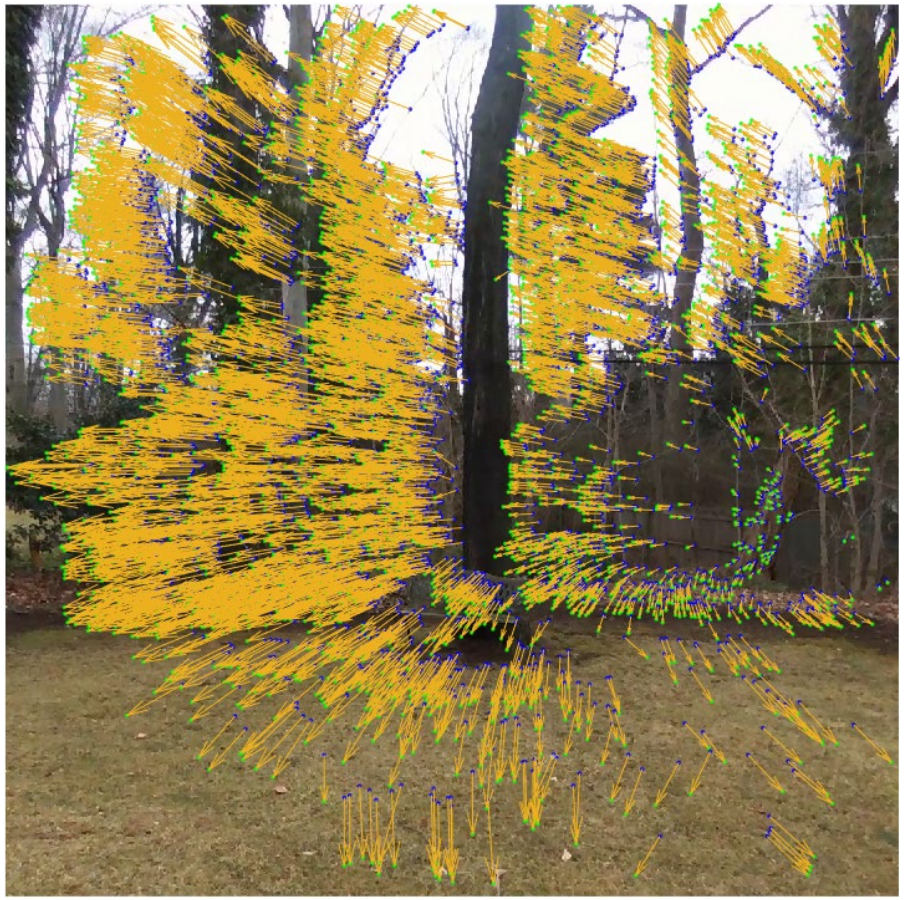
For some subset of patches in a scene, we want to compute how they move.

Lucas-Kanade Algorithm: Nice linear equation connecting spatial gradients, flow, and temporal change:



Solve as $(A^T A)^{-1} A^T b$

$I_t(x, y)$ → $I_{t+1}(x, y)$

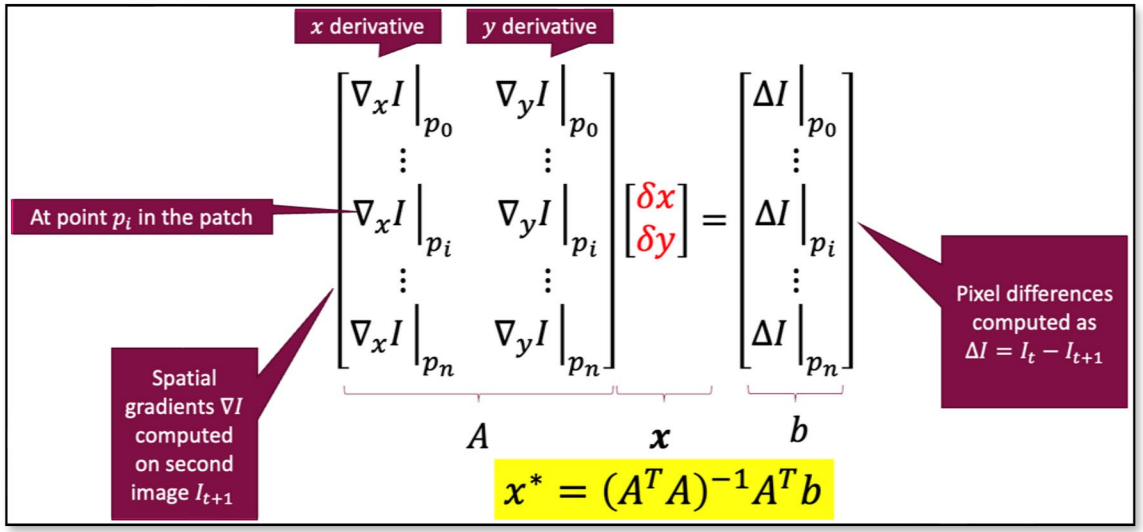


Recap: LK Flow rests on various assumptions

- Brightness constancy
- No Occlusions
- Small and locally uniform motions

But the most important: **Invertibility of $A^T A$**

Perhaps The Most Important Assumption: Invertibility



We assumed we could invert, i.e. compute $(A^T A)^{-1}$

When would this fail? $(A^T A)_{2\times 2}$ is low-rank!

And what does that mean?

We will see: this means that the patch must be sufficiently 'interesting'

Recall: Rank of $A^T A$ = Rank of A

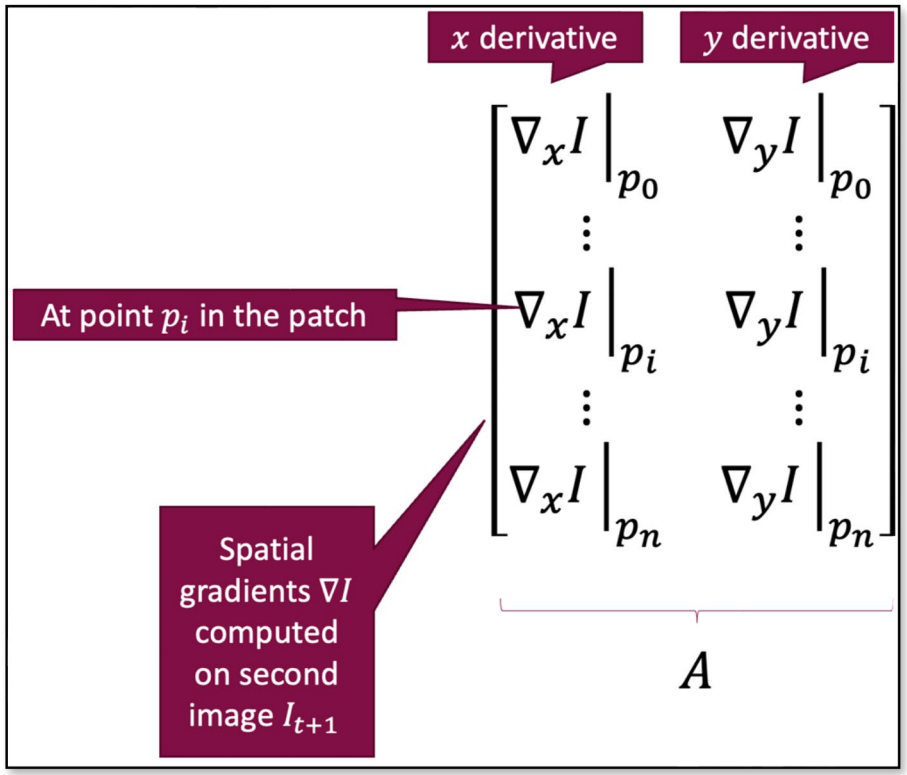
- Easy to see from the SVD:
  - $A = U\Sigma V^T$
  - So, $A^T A = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T$ (since $U^T U = I$)
  - So, eigenvalues of $A^T A$ are square of singular values of A
  - So, number of non-zero eigenvalues of $A^T A$ = number of non-zero singular values of A
  - So rank of $A^T A$ = rank of A

All of this holds for any matrix A. Now, what does this mean for our specific setting?

The Rank of the Spatial Gradients Matrix A

- Low-rank $(A^T A)_{2\times 2}$ means that the $\text{rank}(A^T A) < 2$.
- So, $\text{rank}(A) < 2$.
- Every row is the spatial gradient at a point in the patch.

Rank k means that there are k linearly independent rows in the spatial gradient matrix.

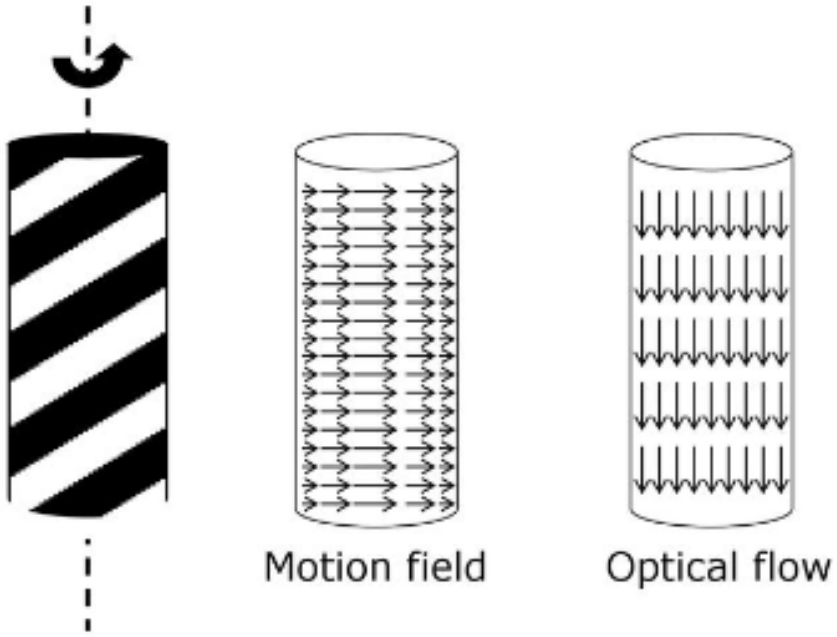


Low-Rank A

Rank(A)=0

- This means that there are *no* linearly independent rows in the matrix.
- This can only happen when the matrix is all zeros.
- i.e. spatial gradient are all zero!

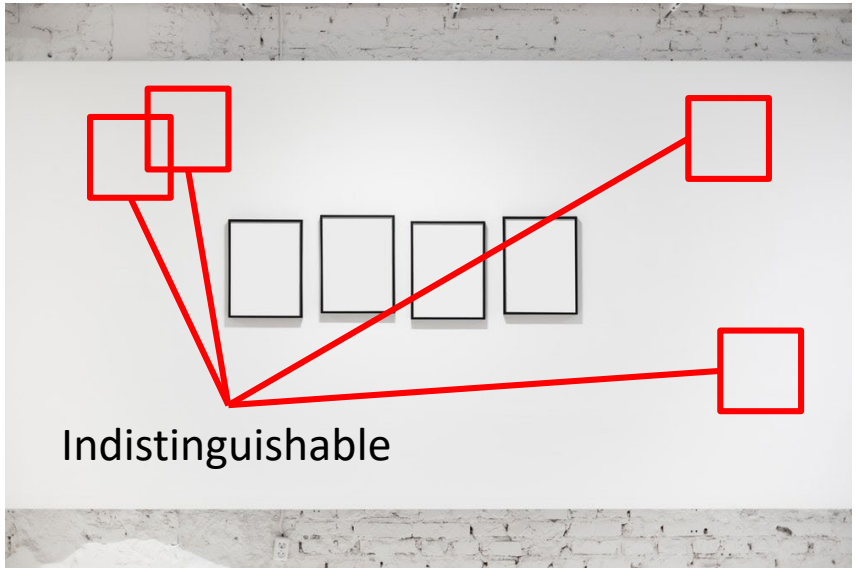
Barber Pole Illusion: Illustrating Rank Deficiency in $A^T A$



Rank 0: The White Wall Problem

$$\nabla I_{t+1}(x, y) = 0$$

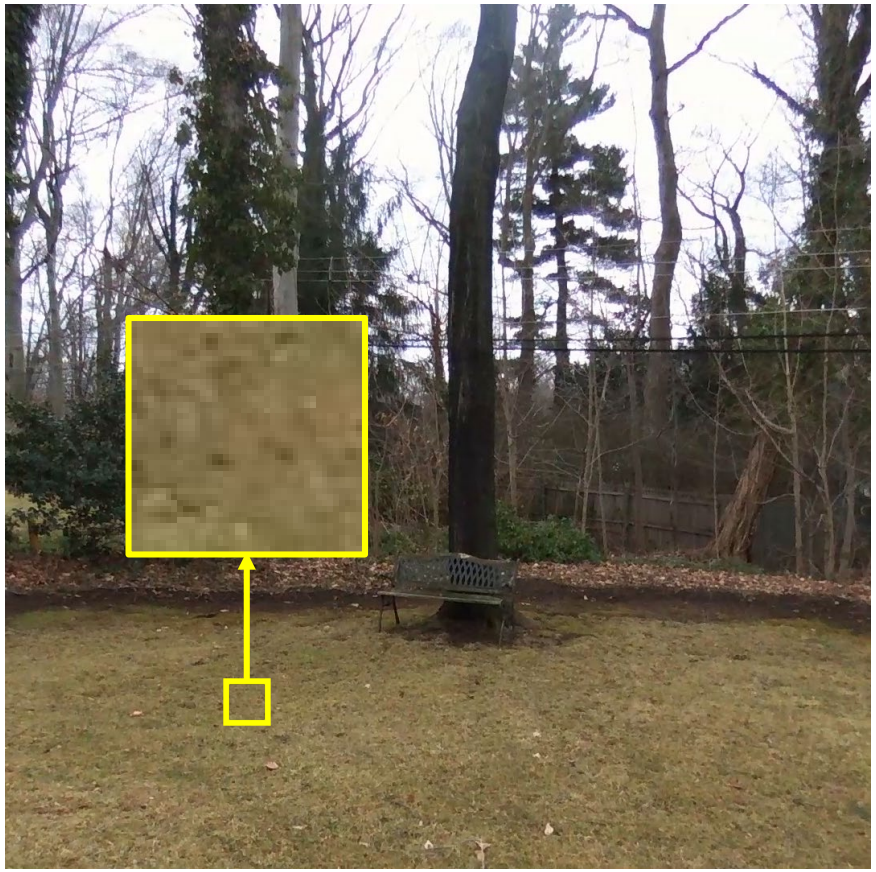
No gradient / close-to-zero gradient = difficult to match.
This is the ‘White wall problem’

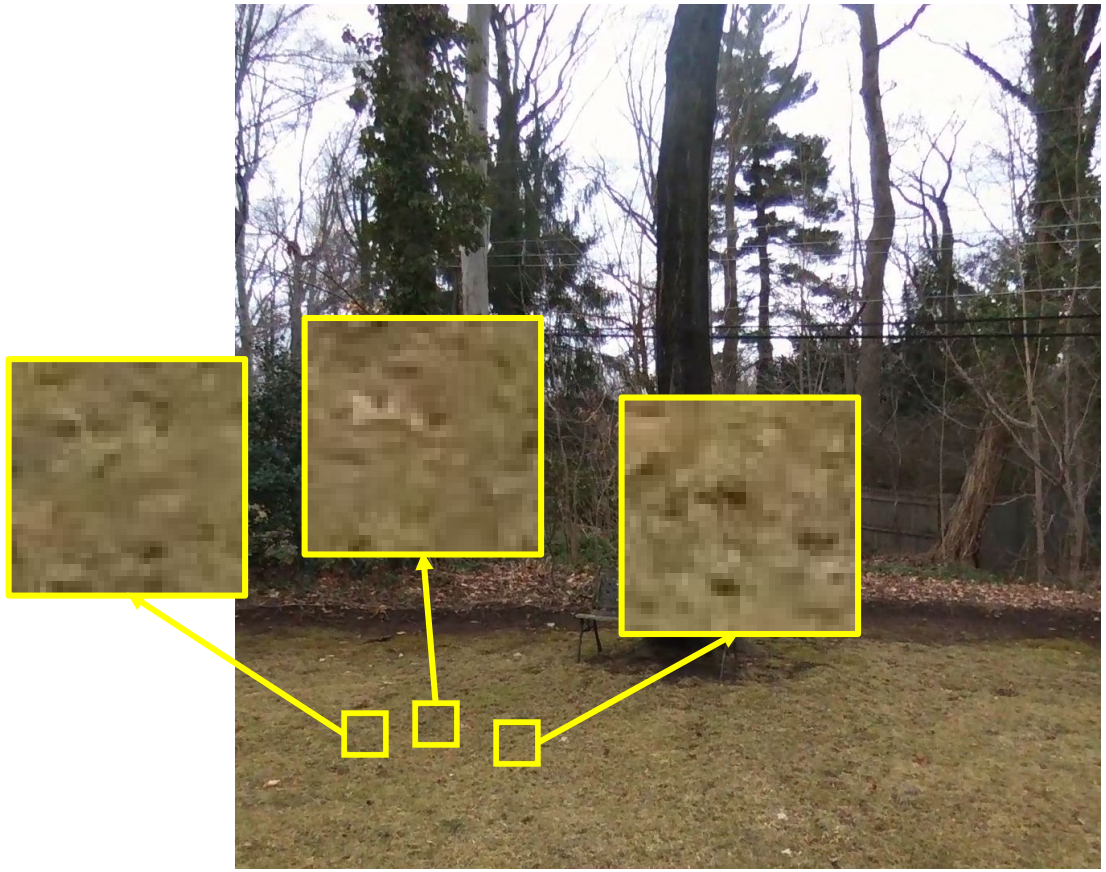


“White Wall” / Flat Patches In Our Yard Example

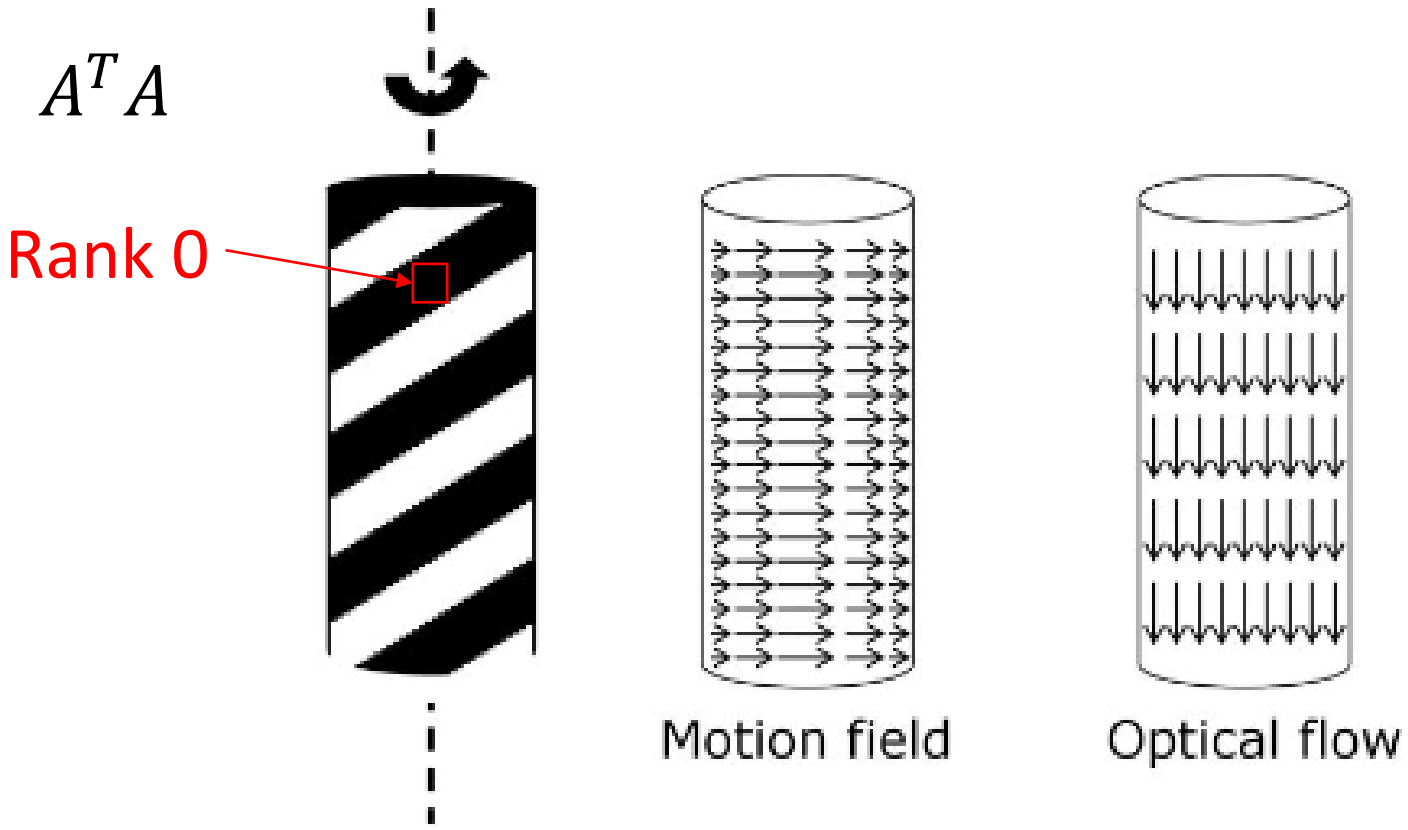
These patches don’t have strong gradients.

Eigenvalues of $A^T A$ are both small. Rank ~ 0





Flat “Rank 0” Patches in Barber’s Pole



Low-Rank A

Rank(A)=0

- This means that there are *no* linearly independent rows in the matrix.
- This can only happen when the matrix is all zeros.
- i.e. spatial gradient are all zero!

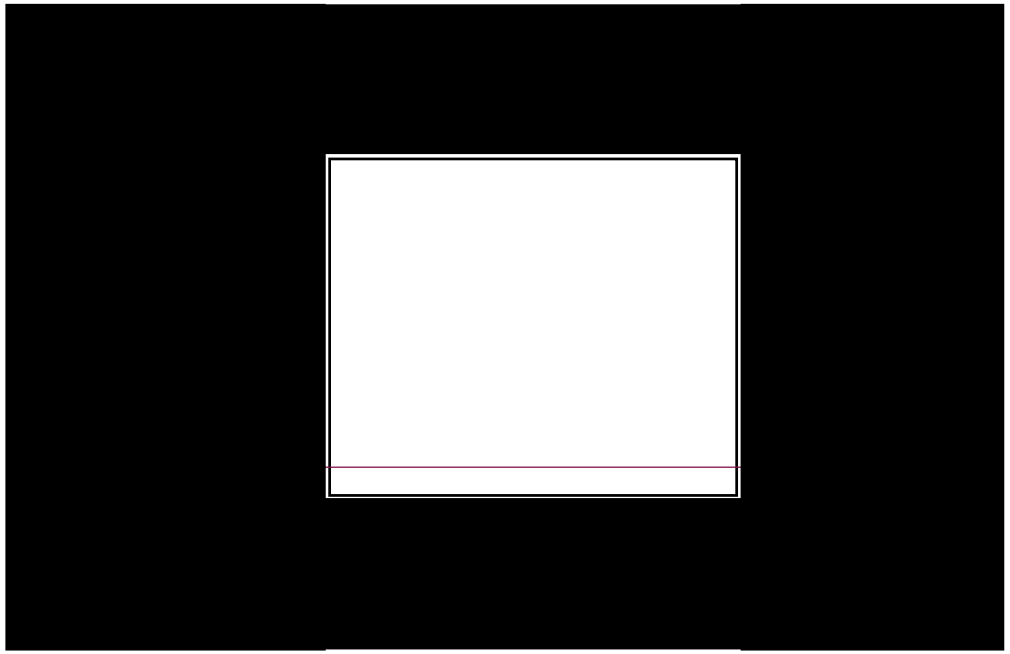
Rank(A)=1

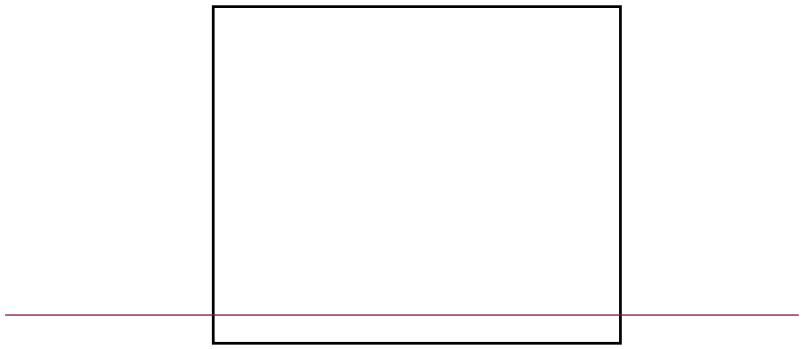
- This means that there is exactly one linearly independent row in the matrix.
- This means all the other rows are multiples of that one row.
- In other words, the gradients are all aligned. (don’t have to be constant)

Rank 1: Uniform Shading / Constant Gradients

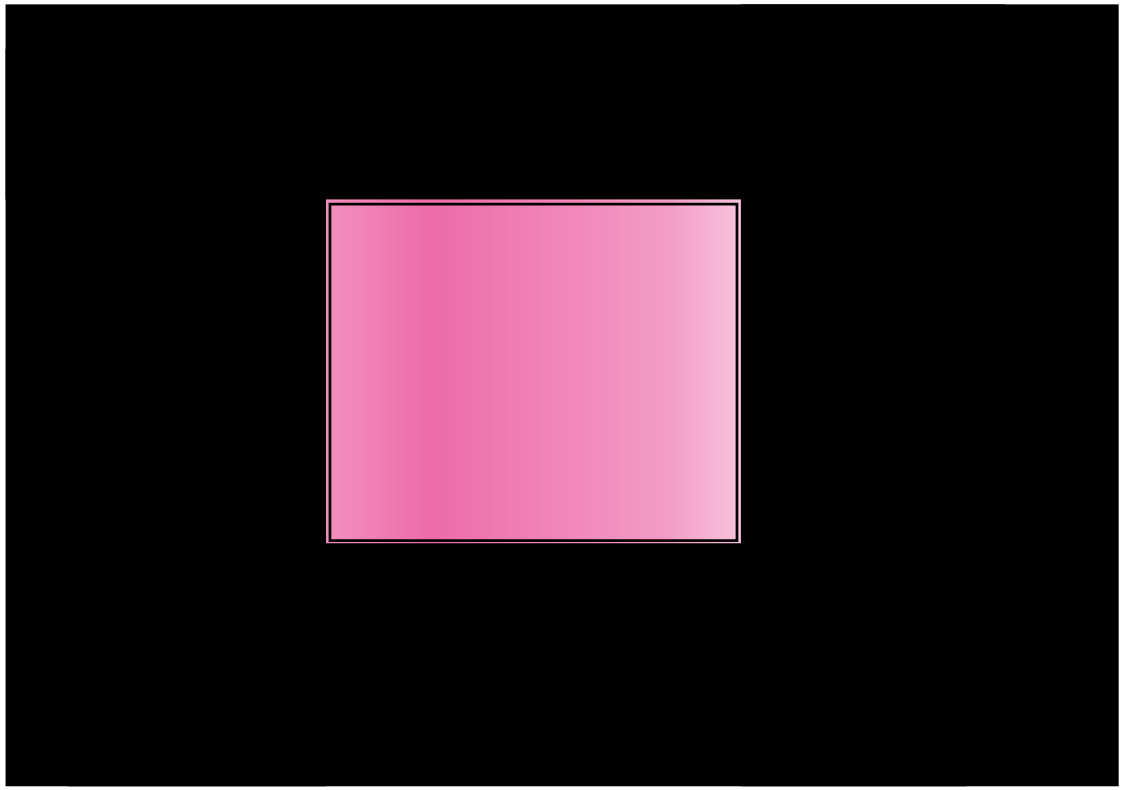
$$\nabla I_{t+1}(x, y) = \vec{c}, \forall x, y$$

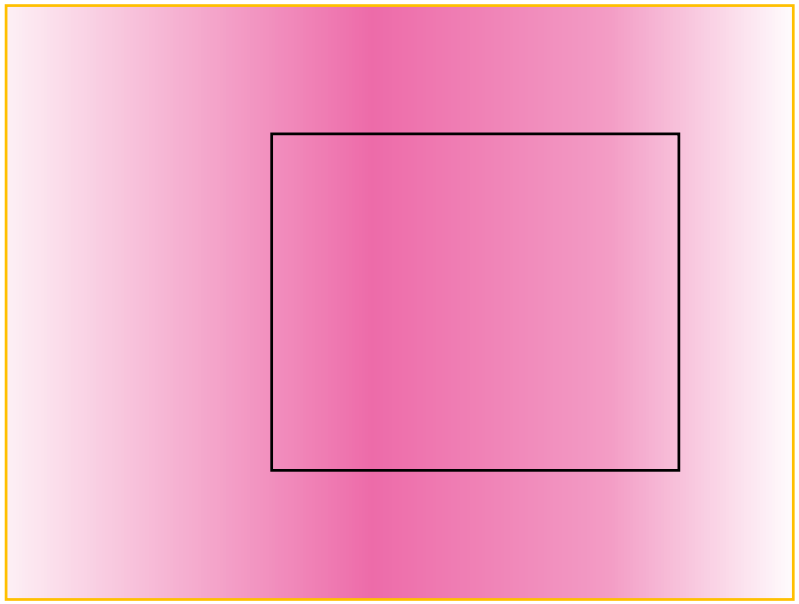
All the (non-zero) gradients in a region are aligned.
No information along direction perpendicular to gradient
Often happens in patches that are flat everywhere except for one contour





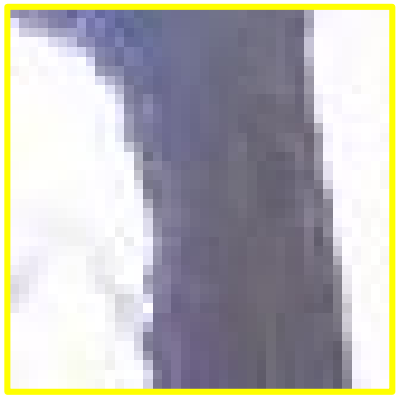
A patch with “aligned” gradients

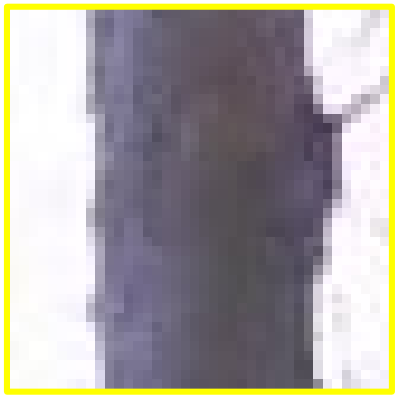


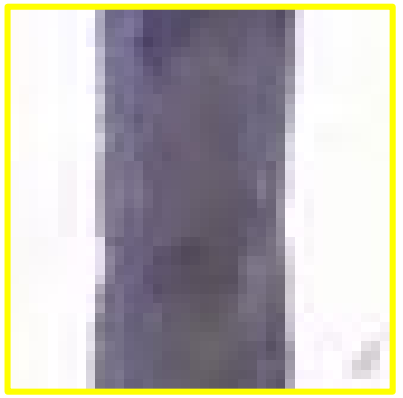


Just like the line, the patch with all aligned (but not *equal*) gradients is also rank 1, and only permits seeing flow *in the direction of the gradient.*

(Approximate) Rank 1 Regions in Our Yard Example





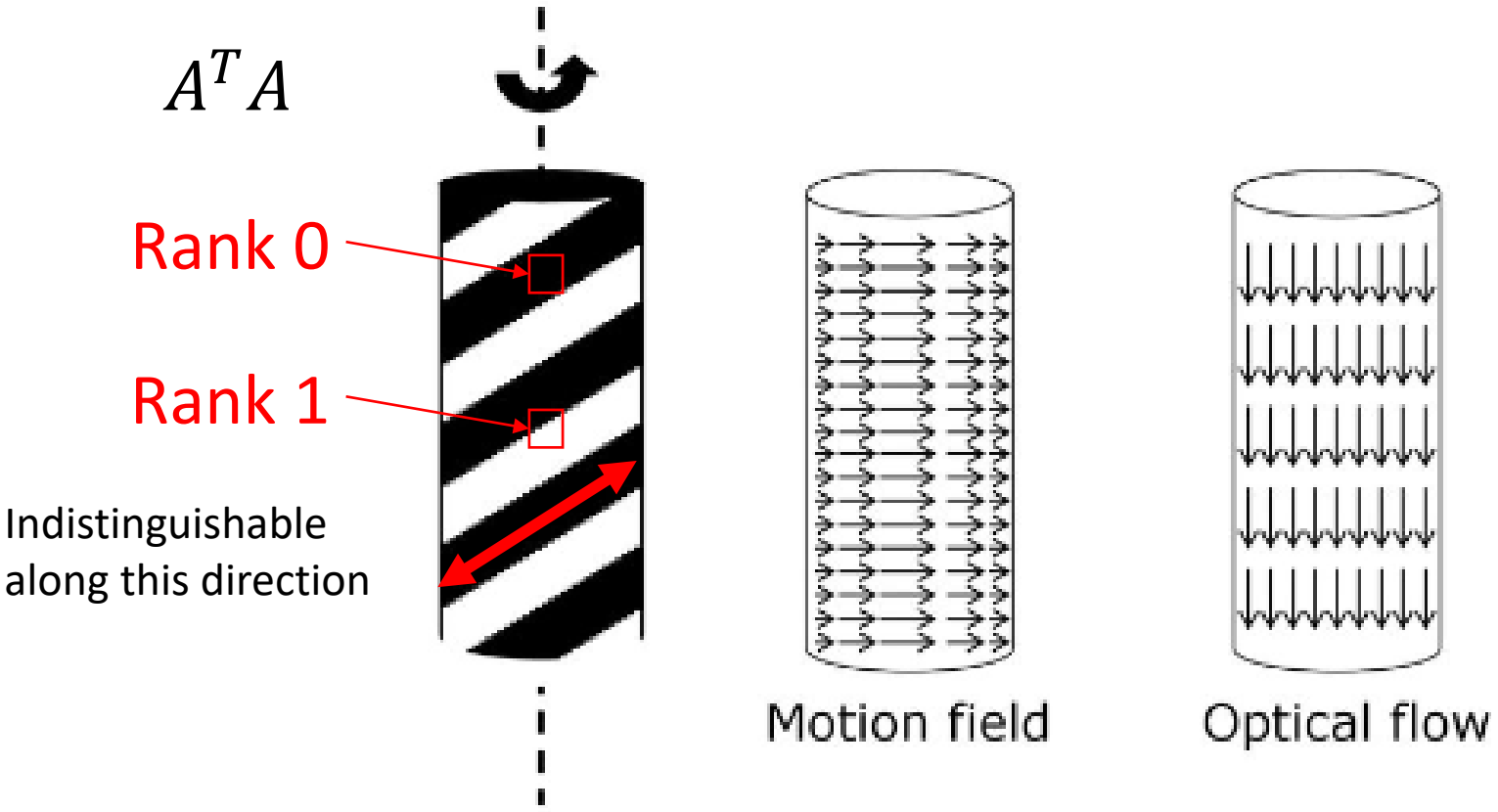


For our example, how high along the tree are we? Eigenvalues of $A^T A$ at these patches are one very large, another very small.

Rank $\sim$ 1

Rank 1 Regions in Barber's Pole

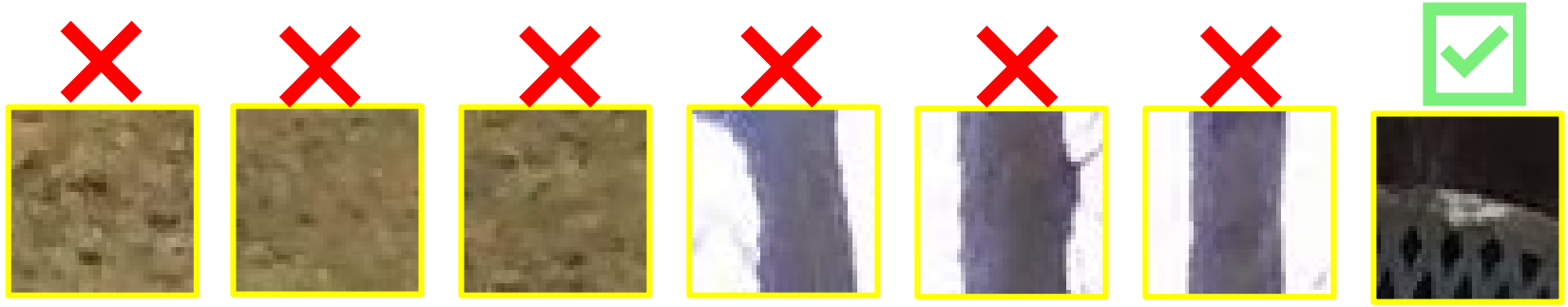
So are there any regions in barber's pole that allow flow computation through invertible $A^T A$? (Back to this in a few slides)



Consequence of Uninvertibility of $A^T A$

- Most places in the image are ‘uninteresting’ – we can’t track them – the interesting places are ‘sparse’.
- Flat regions are bad, edges are bad.
- “Corners” and high-texture regions are good.

Need to find such “features” that are easily trackable.



Good Features / Corners

There is a large literature on this spanning decades, and it typically involves the eigenvalues of $A^T A$.

Module on this later in the course if time permits.

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

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Shape and Motion from Image Streams: a Factorization Method—Part 3

Detection and Tracking of Point Features

Technical Report CMU-CS-91-132

Carlo Tomasi Takeo Kanade

April 1991

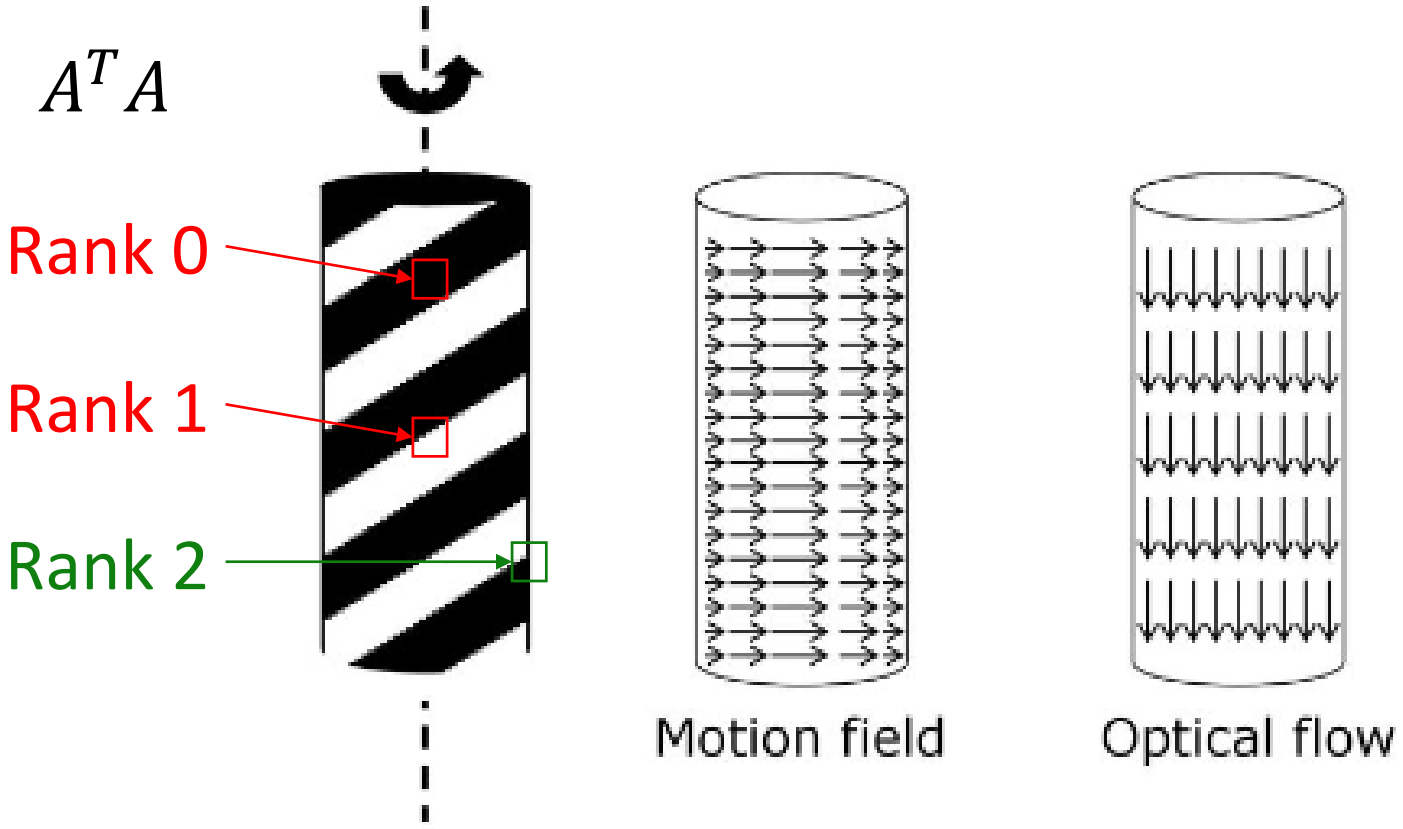
Good Features to Track

Jianbo Shi
Computer Science Department
Cornell University
Ithaca, NY 14853

Carlo Tomasi
Computer Science Department
Stanford University
Stanford, CA 94305

Corners in Barber's Pole

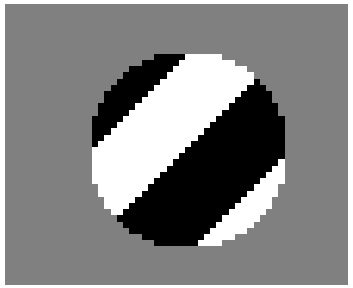
So are there any regions in barber's pole that allow flow computation through invertible $A^T A$?



https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html

Your brain looks for corners too!

Inside the barberpole, where there are no corners, only edges, flow actually looks perpendicular to stripes.



It’s only on the rank 2 “corner” patches at the boundaries of the pole that it looks vertical.

Thus, the fact that the flow looks vertical to you is evidence that you are relying on these corner points!

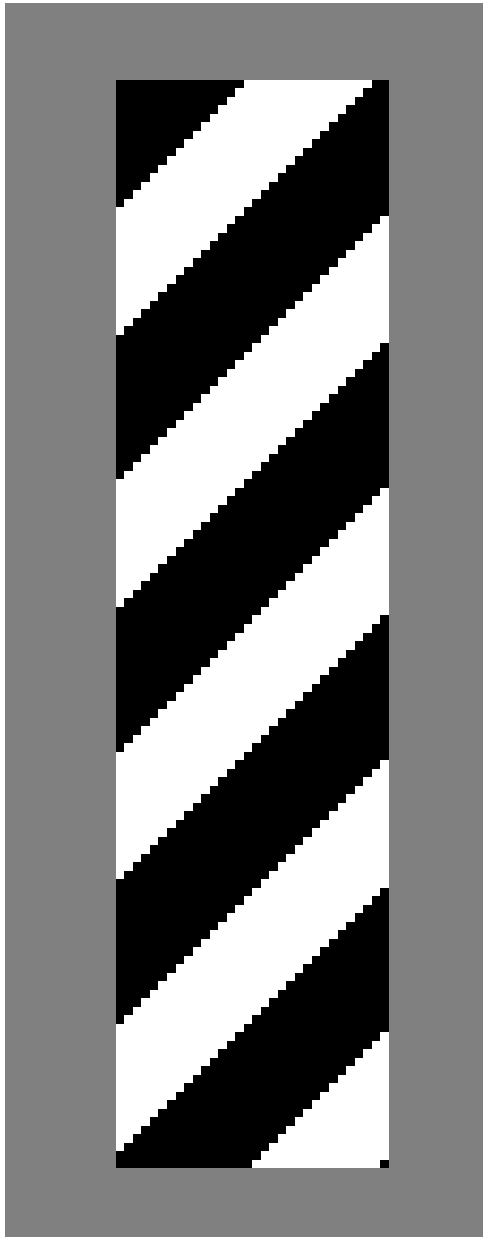


Figure: Wikipedia

Optical Flow Confidence

- Check determinant of $A^T A$ (2x2 matrix) for invertibility.
- Can rank various pixels according to determinant, or according to smallest singular value of A = sqrt of smallest eigenvalue of $A^T A$
  - Smaller determinant or smaller eigenvalue => closer to being uninvertible => low "confidence"

- Also possible for b to not be in the range of A, i.e., the minimum MSE solution x^* produces high MSE $||Ax^* - b||_2^2$
  - Can happen if assumptions are violated, e.g., the light goes off, or an object became occluded, shiny surface etc.

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

$A \qquad x \qquad b$

Extensions

There is a great deal of research trying to avoid these assumptions:

- Compute flow bi-directionally for better reliability
- Assume affine deformations / other “motion models”
- Compute the flow on multiple resolutions of the image and iterate to capture larger flow

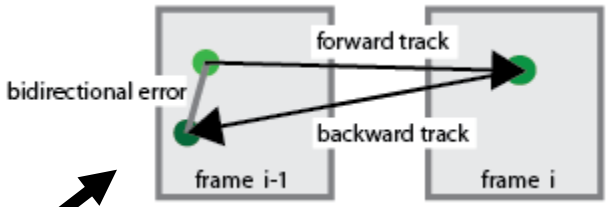
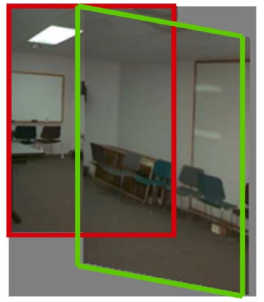


Image from MATLAB site



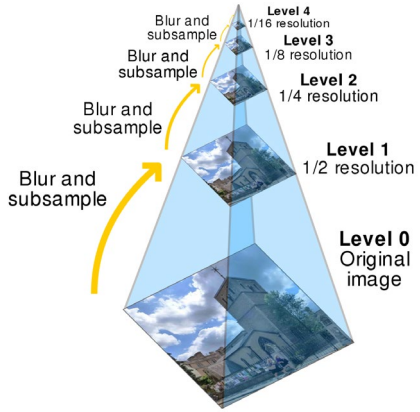
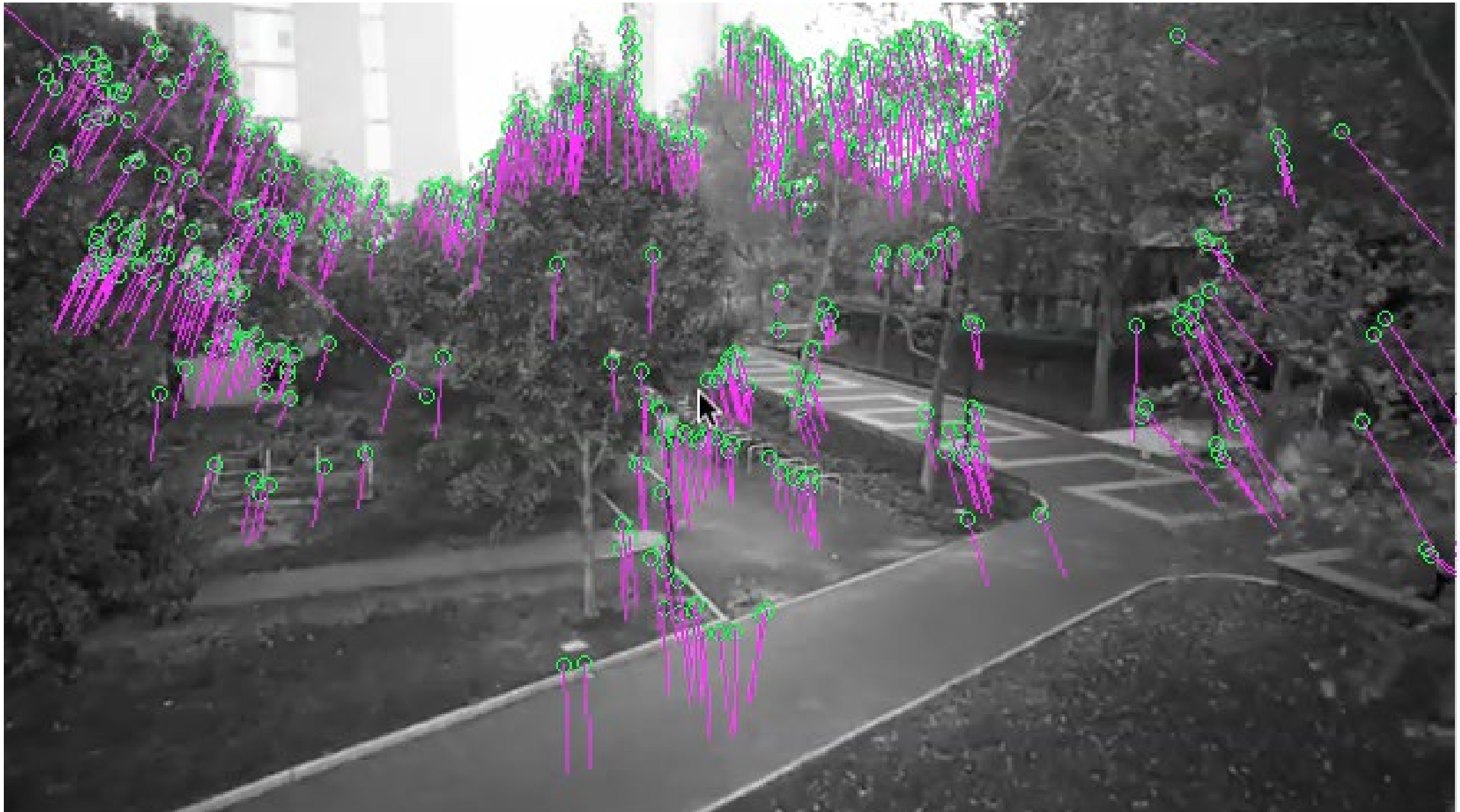


Image from wikipedia

Inferring 3D Motion from Optical Flow

Which direction is the vehicle moving?



Flow Contains 3D Motion Information

Optical Flow: 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 equal to motion field (motion of the projected 3D points), but not always, as we have seen.

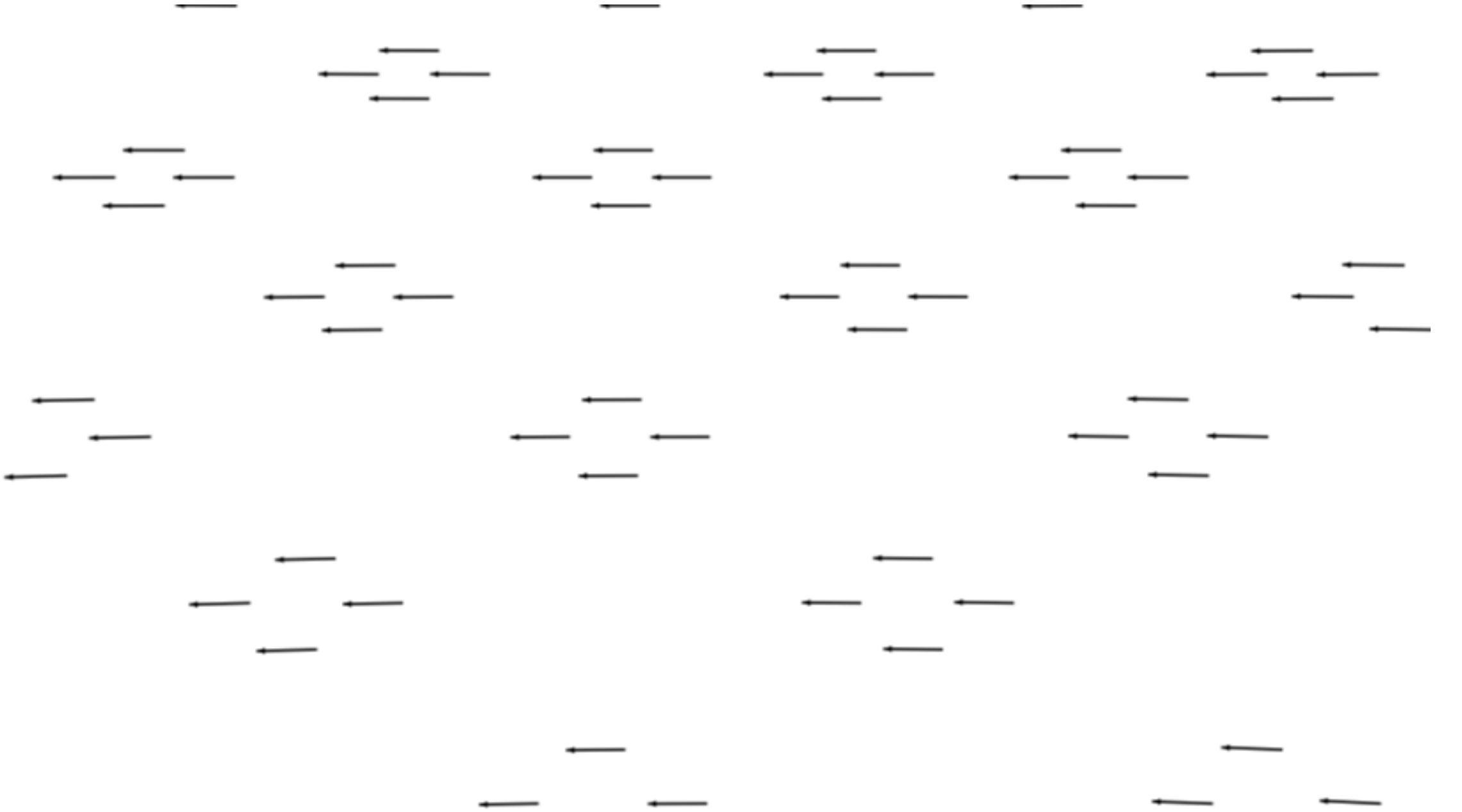
If we assume that it is equal to the motion field, then how does the optical flow field look like for various camera-world relative 3D motions in a static world?

Pure translation

Turning towards the right

Pure horizontal translation

Pure rotation around vertical axis



The Problem of 3D Motion from Optical Flow

We want to answer precisely:

- How does the image of a point move when the camera moves? (or equivalently, a static world moves relative to the camera)

- Could we use optical flow help compute the 3D camera motion?
  - Recall: optical flow is not quite a motion field, but we will have to treat it as such, in this pursuit.

Note: 2-View SfM Already Solves for “Motion”

If we know nothing about the camera motion and have no additional information, inferring motion = 2-view SfM problem, specifically the “motion” part of structure from motion!

- Recall the solution: Epipolar constraints eliminate the depth, then we solve for E, then get camera motion R, T. Done!*
  - *In SfM, at this point, we also wanted to recover depth, so we triangulated.

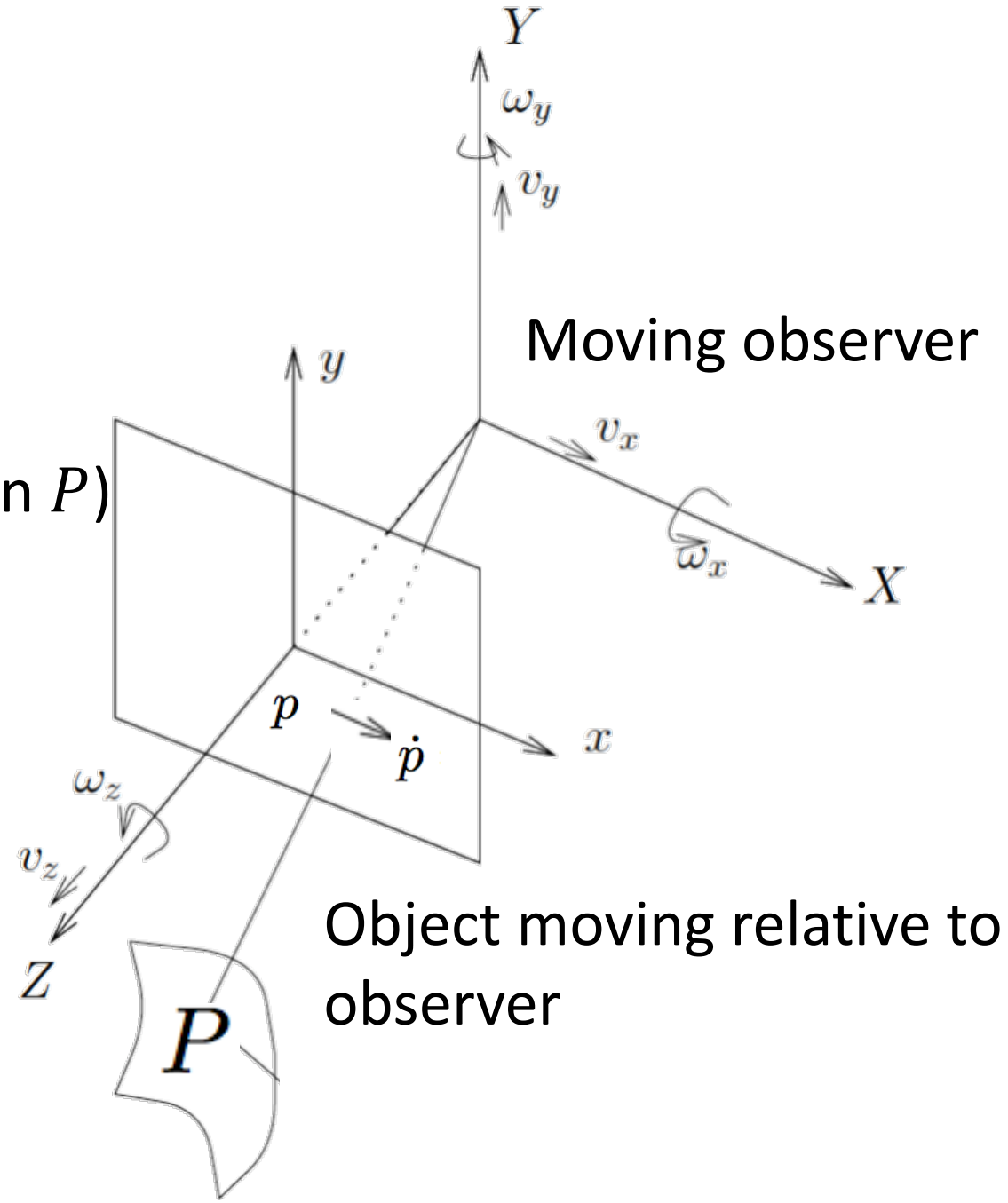
- In this module, we instead address simpler settings where more information is available to us and show more efficient methods.
  - In particular, R is known, or depths known (both reasonable for robots and one for animals too)

Projection equations for calibrated camera

$x = \frac{X}{Z}, y = \frac{Y}{Z}$

or in vector notation $p = \frac{1}{Z}P$

(homogeneous $p = (x, y, [1])$, Euclidean P)



Equation 1: pixel motion in terms of 3D object motion

Projection equations for calibrated camera

$x = \frac{X}{Z}, y = \frac{Y}{Z}$

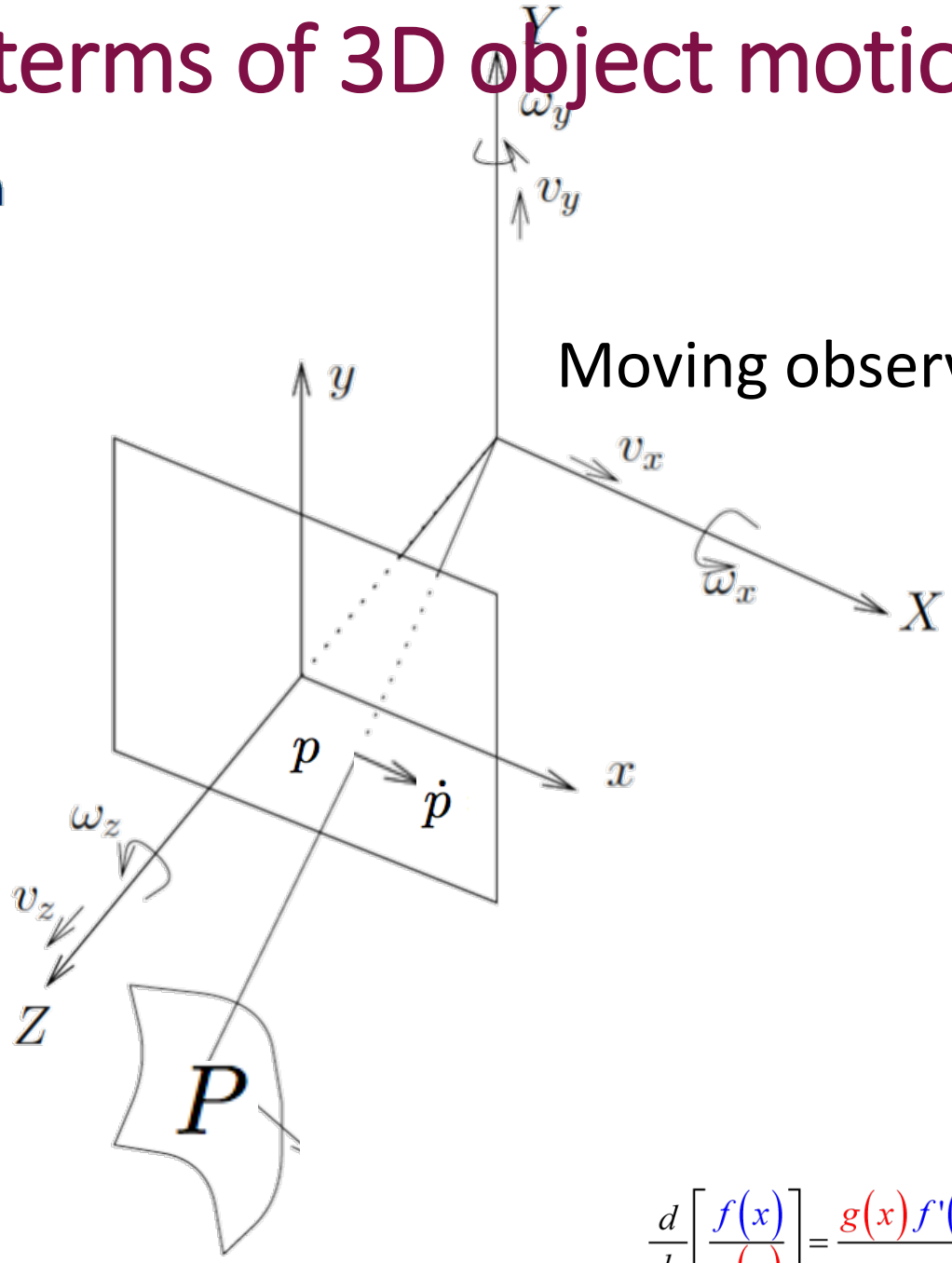
or in vector notation $p = \frac{1}{Z}P$

Differentiating w.r.t. time yields:

$$\dot{p} = \frac{\dot{P}}{Z} - \frac{\dot{Z}}{Z}p$$

Velocity of projection

Projection of 3D velocity



Moving observer

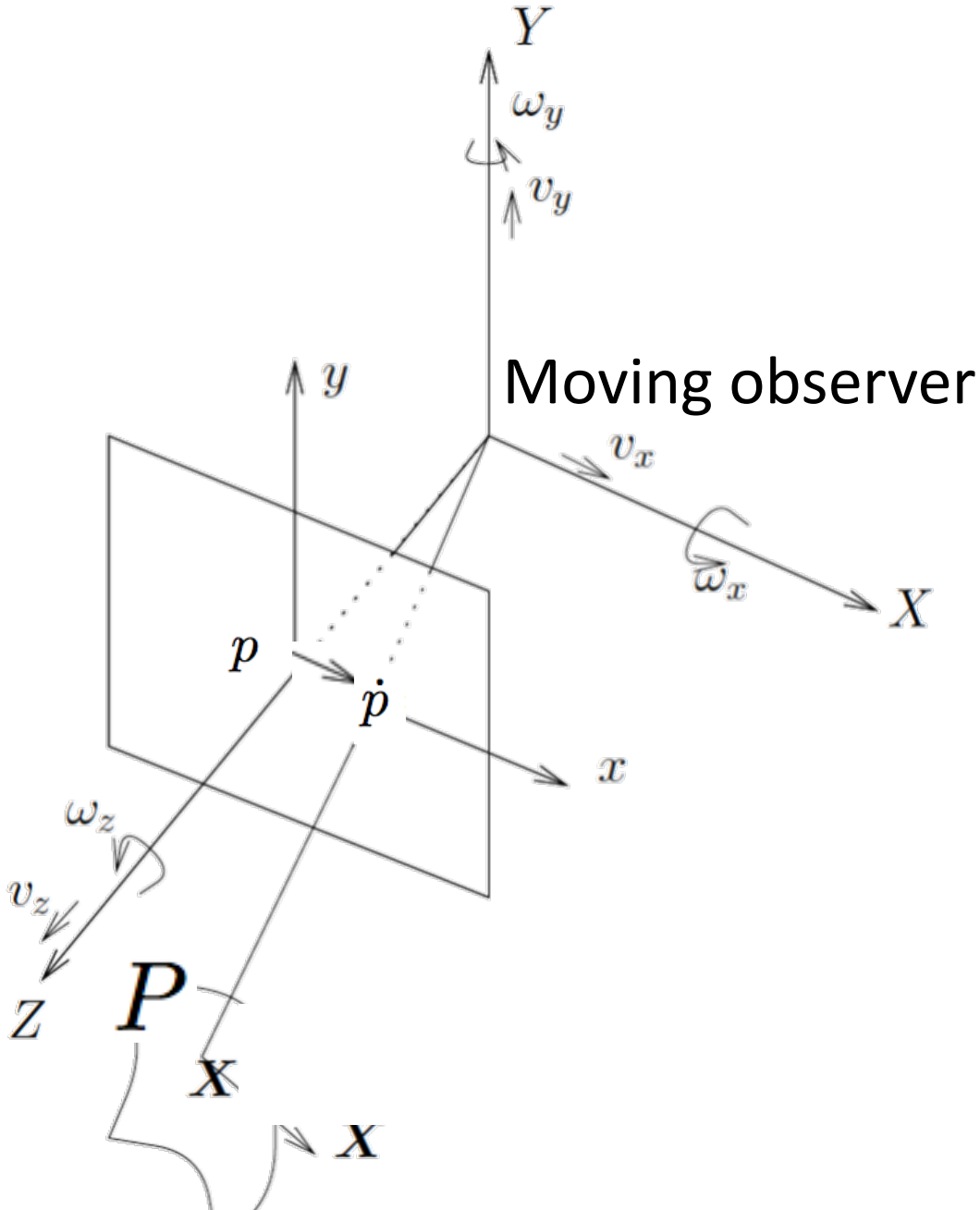
$$\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{\left(g(x)\right)^2}$$

Equation 2: 3D object motion in terms of camera motion

For a camera moving at velocity $\boldsymbol{V}$
spinning at angular velocity $\boldsymbol{\Omega}$

All in camera coords

$$\dot{P} = -\Omega \times P - V$$



Combining the two key equations

$$\dot{p} = \frac{\dot{P}}{Z} - \frac{\dot{Z}}{Z}p \qquad \dot{P} = -\Omega \times P - V$$

Notation abuse warning: $p = (x, y, 1)$, but we will sometimes write $\dot{p} = (\dot{x}, \dot{y})^T$

And $V = \left[V_x, V_y, V_z\right]^T$

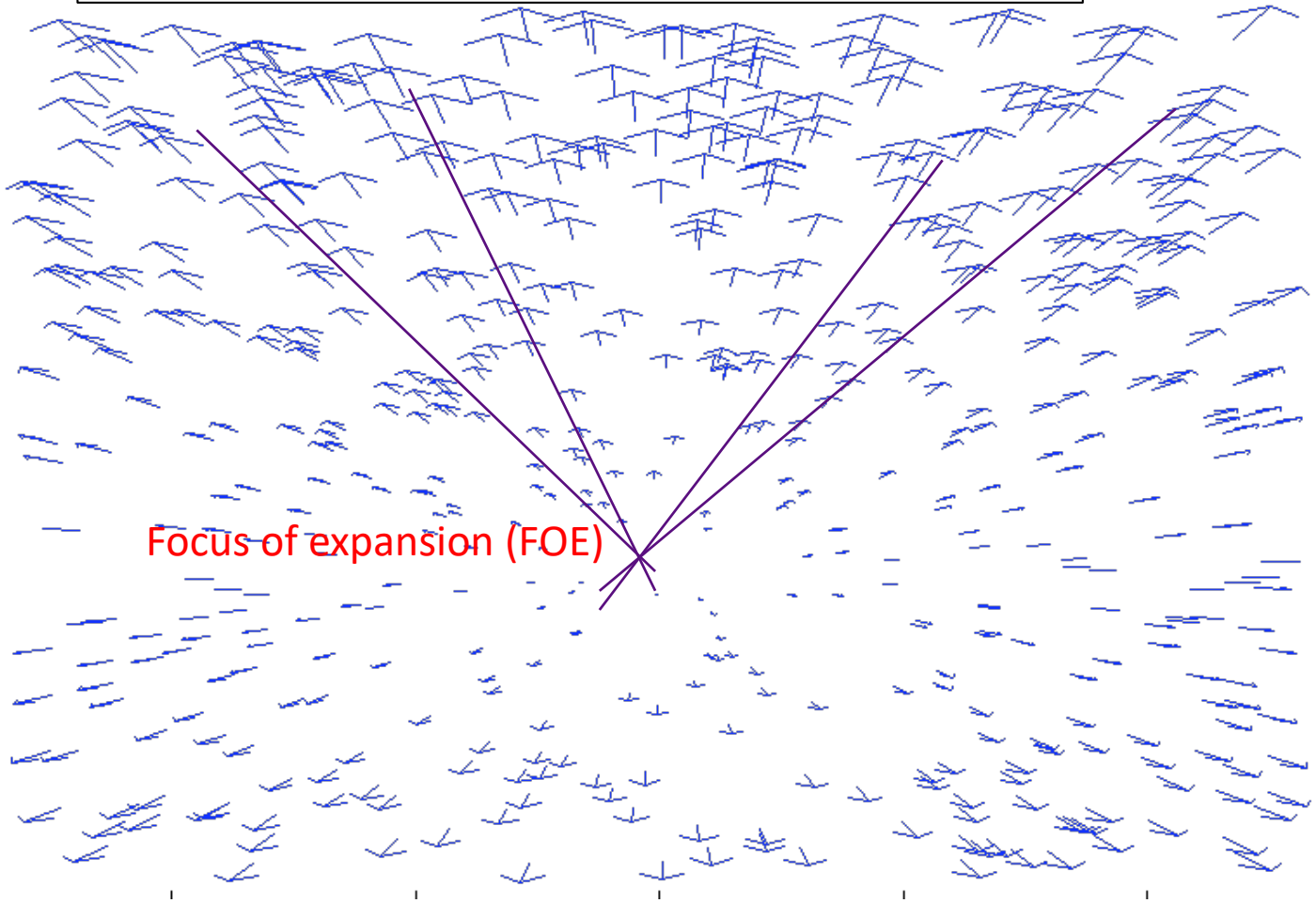
"Left as an exercise to the reader" ;)

$$\dot{p} = \underbrace{\frac{1}{Z}\begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix}}_{\text{translational flow}} + \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix}\Omega}_{\text{rotational flow independent of depth}}$$

Optical flow has two additive components: translational and rotational.

Translational Flow Part 1: Distance from FOE

Q: What must the world look like for this image to be a translational flow map for forward motion?
A: Z must be approx. constant



$$\dot{\boldsymbol{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{V_z}{Z} \begin{bmatrix} x - \frac{V_x}{V_z} \\ y - \frac{V_y}{V_z} \end{bmatrix}$$

Focus of expansion (FOE)

Change in x (or y) coordinate is:

- proportional to how far away that coordinate is from V_x/V_z (or V_y/V_z)

$$\dot{p} = \underbrace{\frac{1}{Z} \begin{bmatrix} xV_z - V_x \\ yV_z - V_y \end{bmatrix}}_{\text{translational flow}} + \underbrace{\begin{bmatrix} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \end{bmatrix} \Omega}_{\text{rotational flow independent of depth}}$$