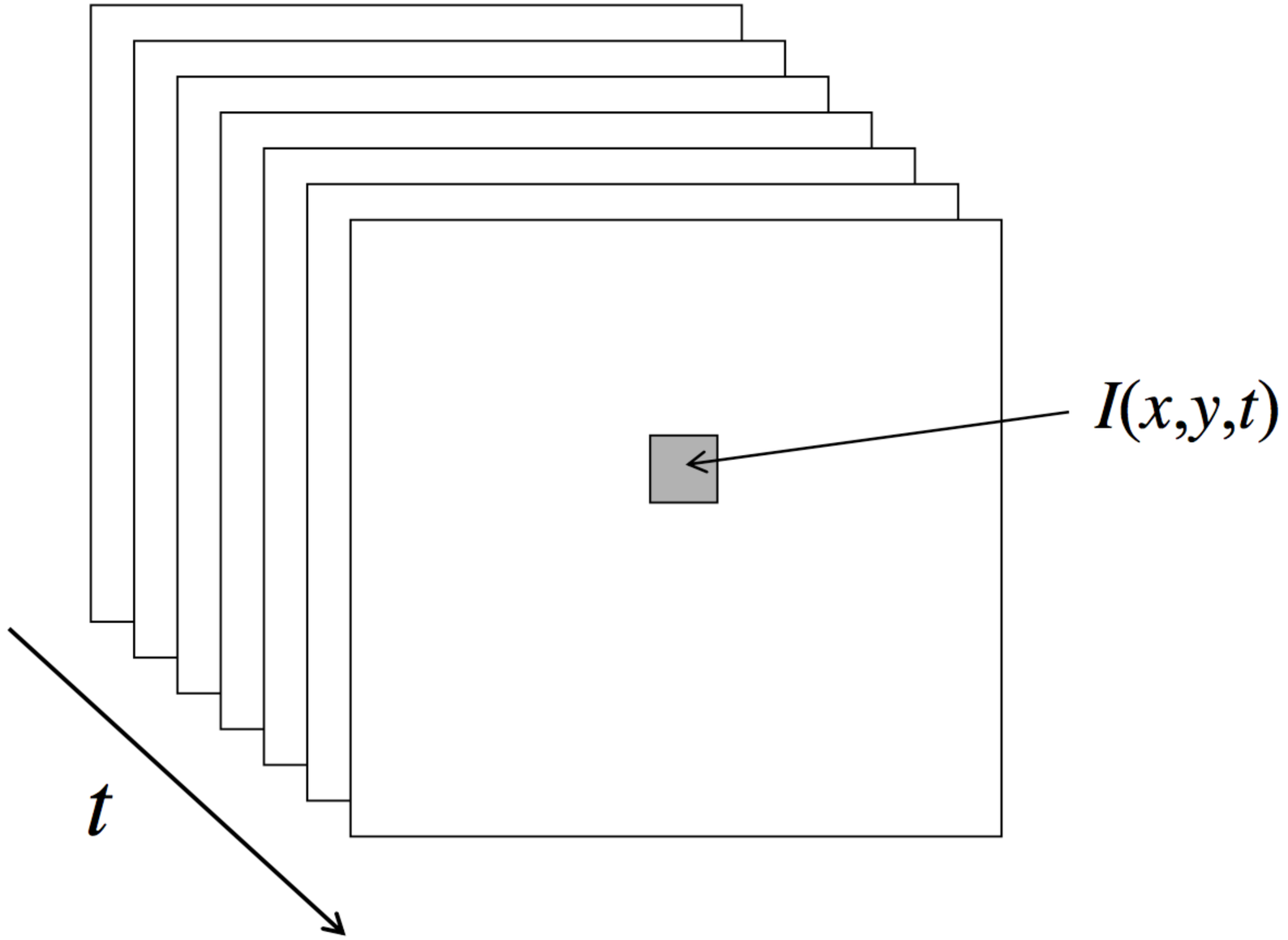
Motion, Tracking & Optical Flow

Gary Overett (Slides adapted from CMU 16-720 2014)

Szeliski Chapter 8

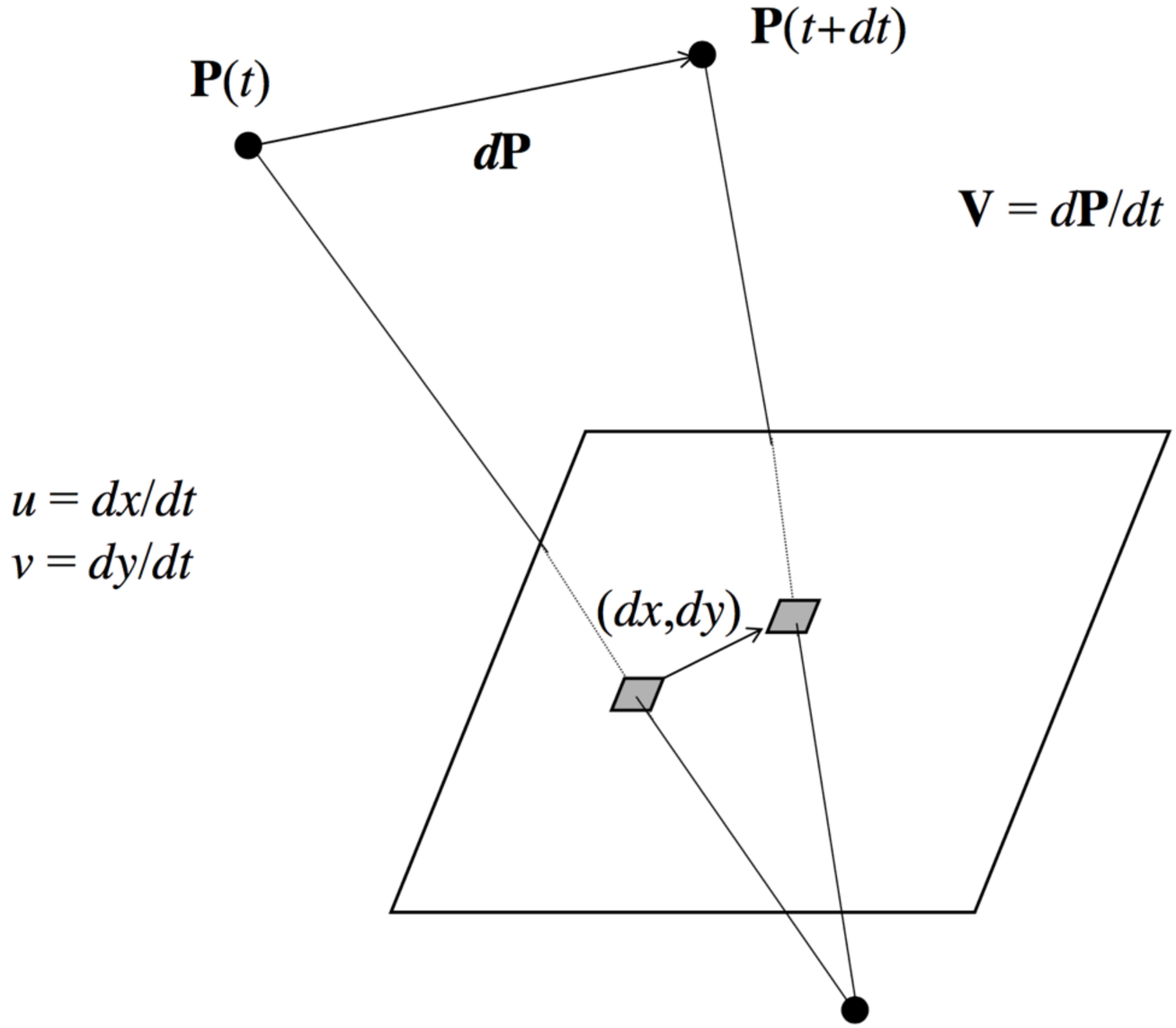

STOP


I(x,y,t)
t

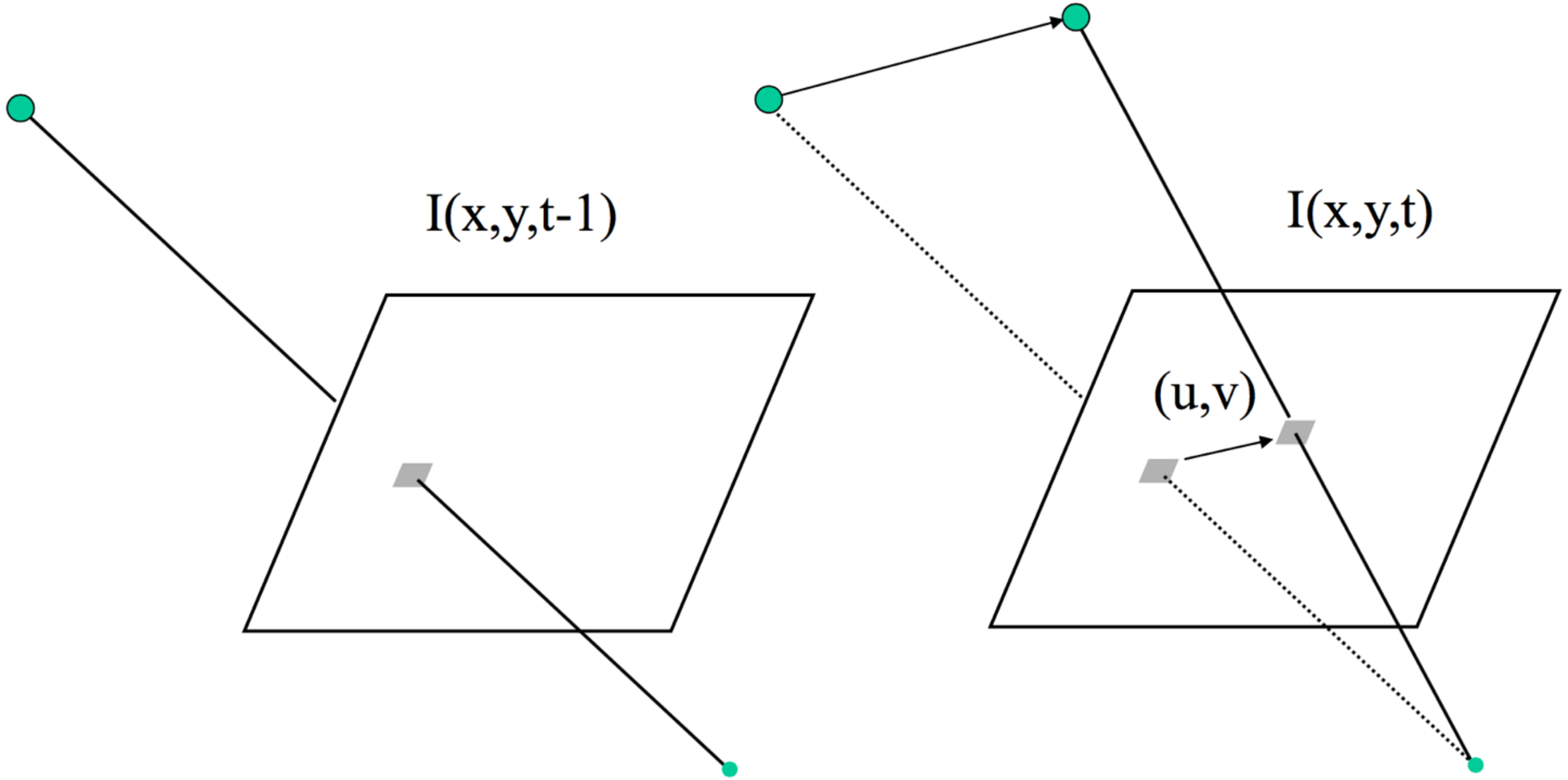
Summary

- Using Gradients to Estimate Motion
- Ambiguities
- Constant Motion Assumption
- Affine Flow Assumption
- Planar Assumption

Motion Field

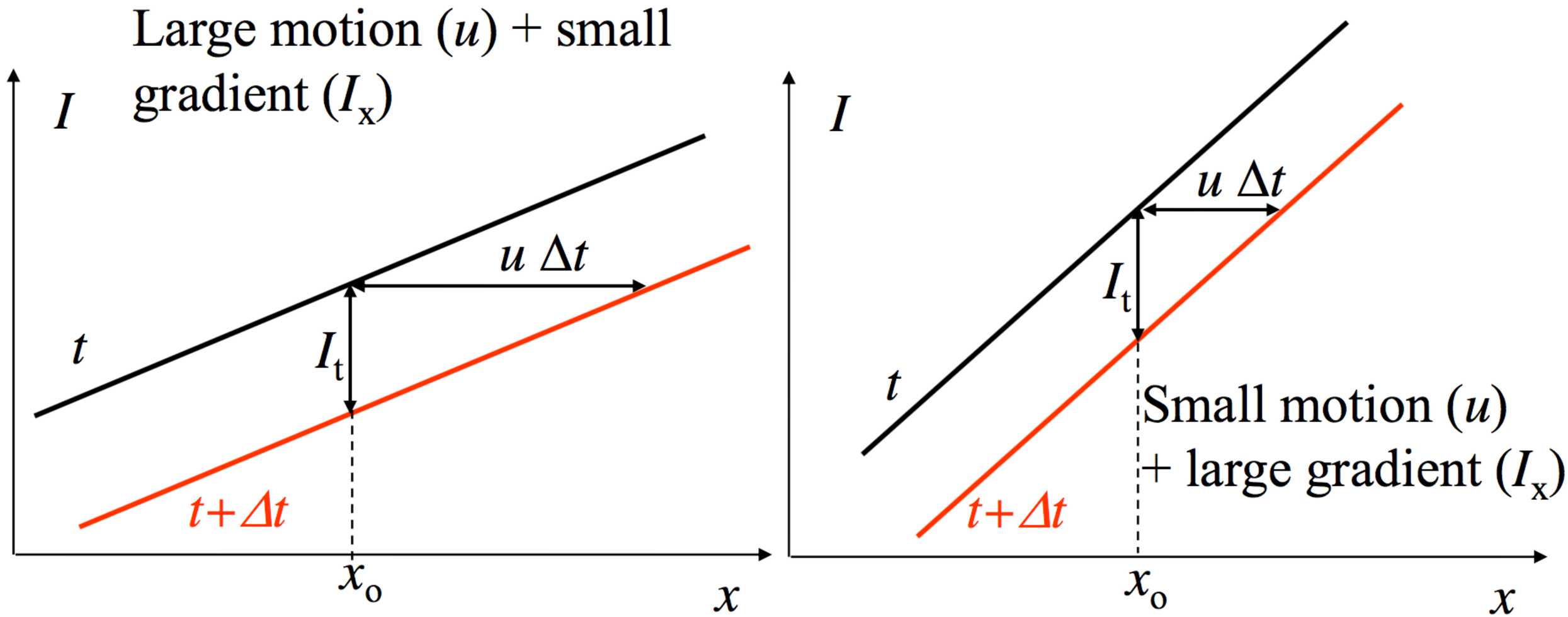


Gradients & Motion



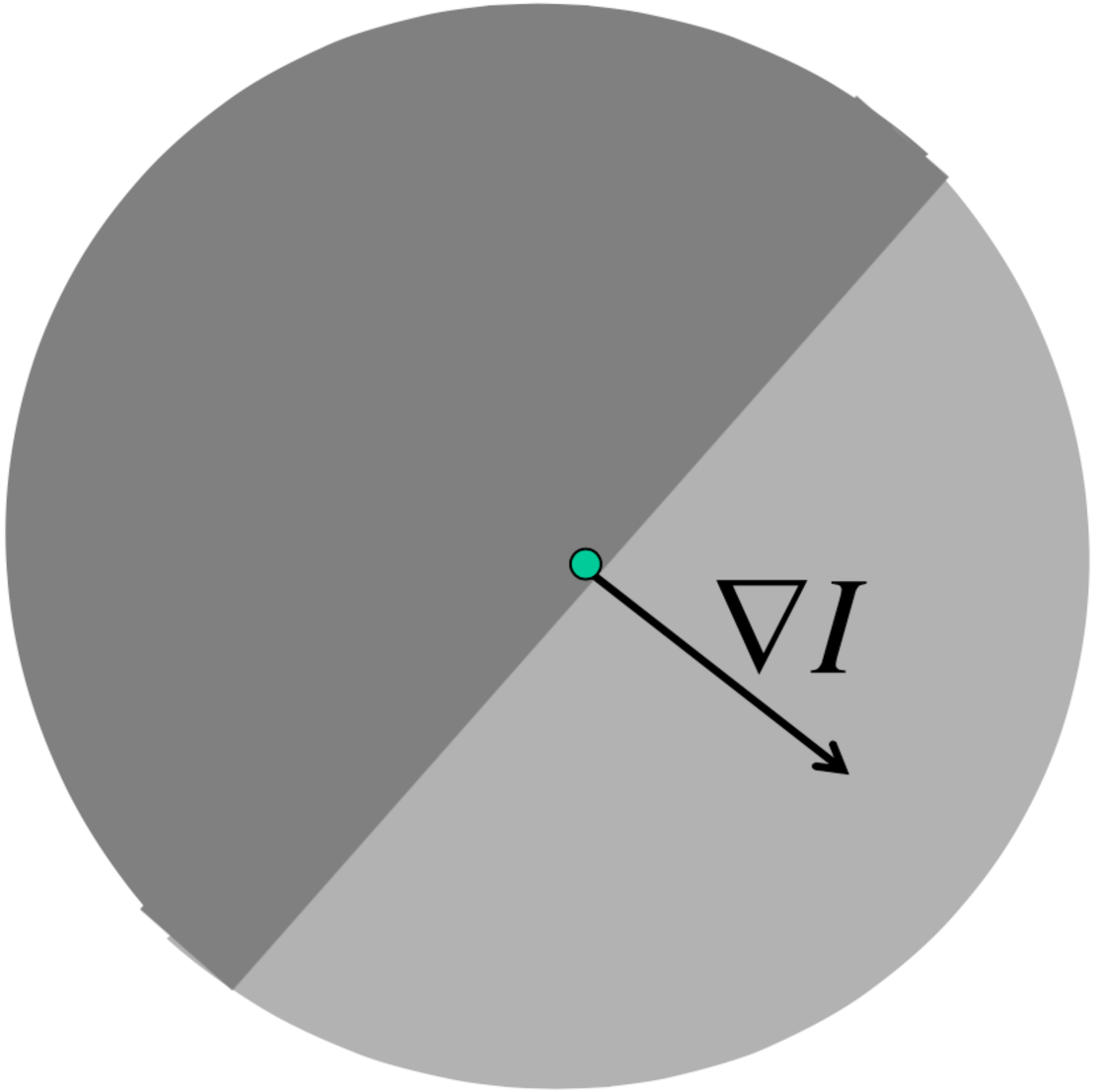
$$uI_x + vI_y + I_t = 0$$

Large & Small Motions

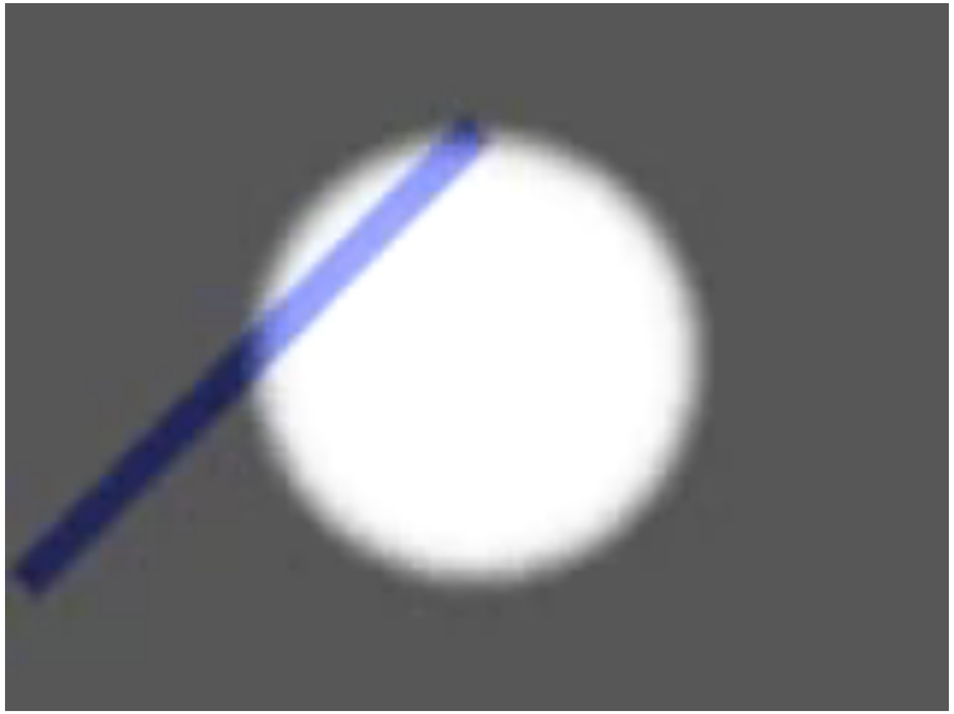


The change of image brightness at x_o, $I(x_o)$, is the same in both cases

Apperture Problem



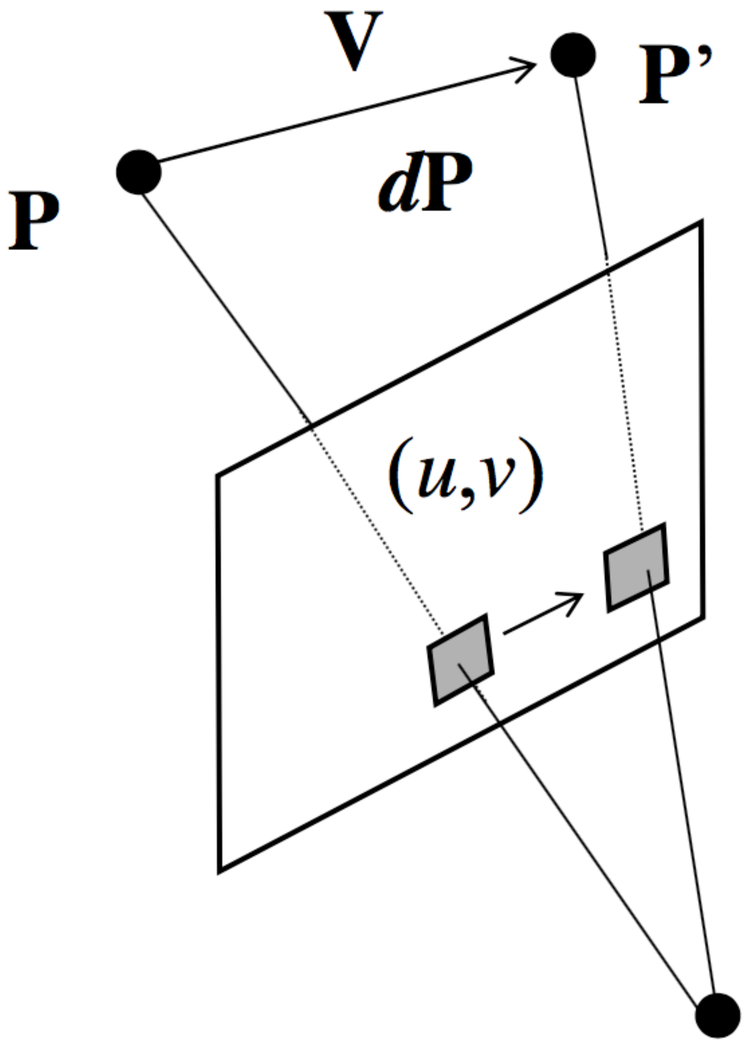
Apperture Problem



Think about it…

- When does the constant brightness assumption fail?
- There are at least 3 distinct cases.

Parallax

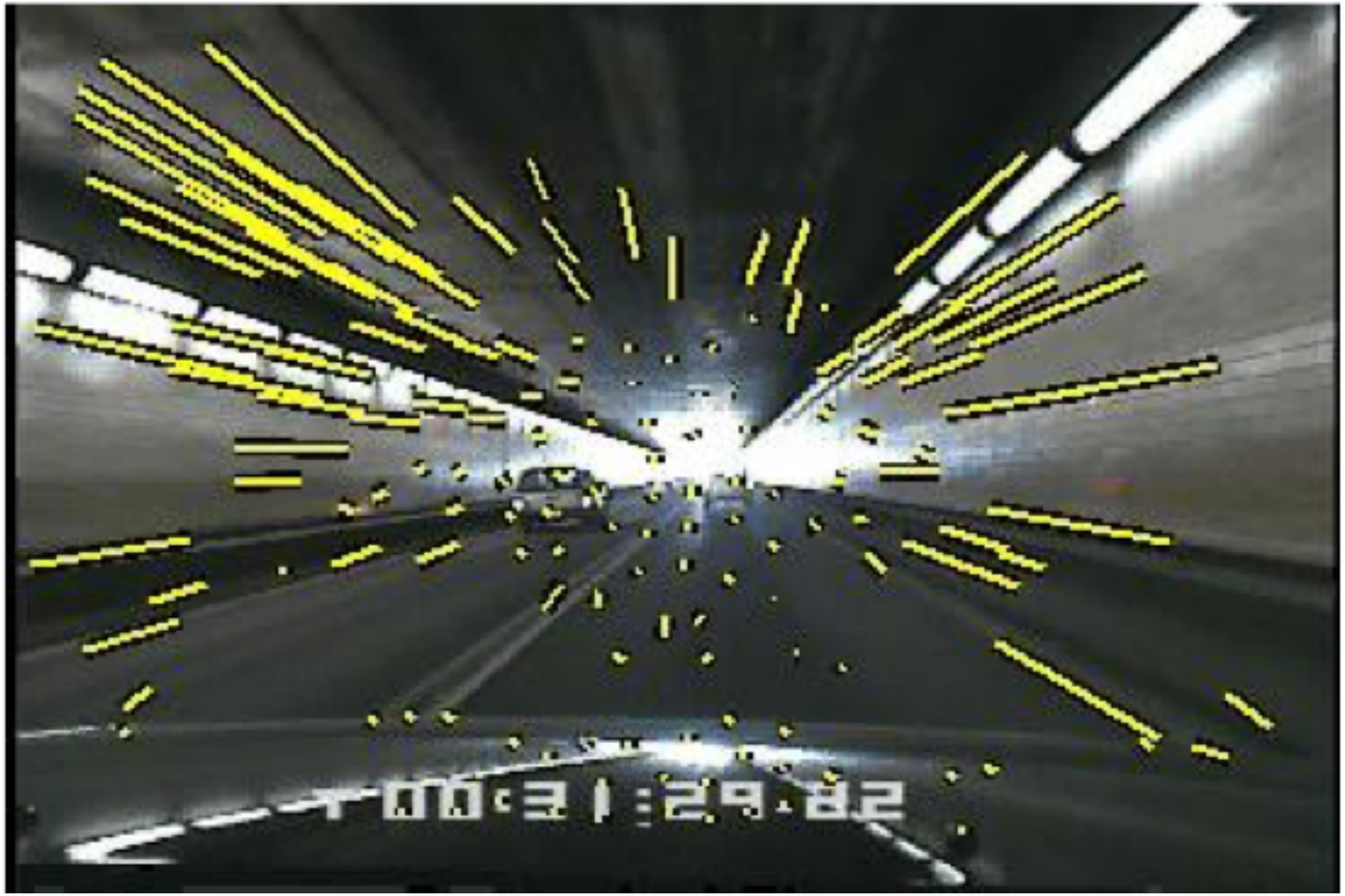


$$u = \frac{V_X - xV_Z}{Z} \qquad v = \frac{V_Y - yV_Z}{Z}$$

Even if we assume translation of all the points in the scene, we still have N+3 unknowns because of the Z.

Translation Only

- All the flow vector intersect at one point the FoE (Focus of Expansion) where there is zero motion
- The flow is greater toward further from the FOE

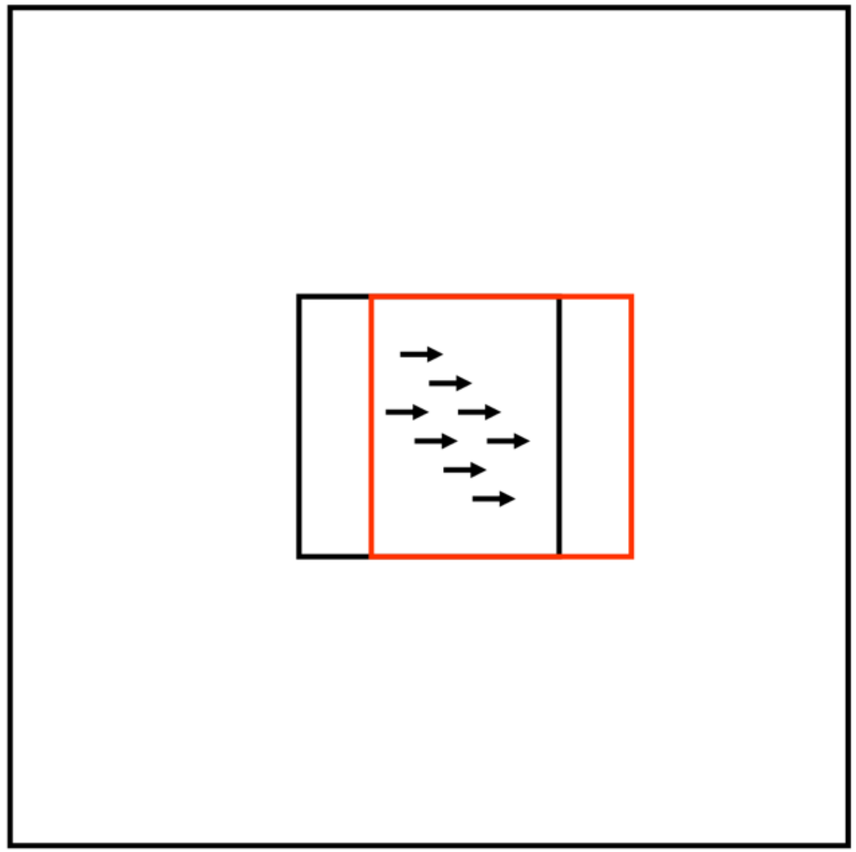


Constant Flow

$$\underset{u,v}{Min} \sum_{x,y \in W} \left(uI_x + vI_y + I_t \right)^2$$

$u(x,y) = u$

$v(x,y) = v$



Constant Flow Case

$$\underset{u,v}{\boldsymbol{Min}} \sum_{x,y \in W} \left(uI_x + vI_y + I_t \right)^2$$

- Linear least square in (u,v) :

$$Min_{u,v} \left\| \begin{bmatrix} I_x & I_y \\ \vdots & \\ I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} I_t \\ \vdots \\ I_t \end{bmatrix} \right\|^2$$

$$Min_{u,v} \left\| A \begin{bmatrix} u \\ v \end{bmatrix} + b \right\|^2$$

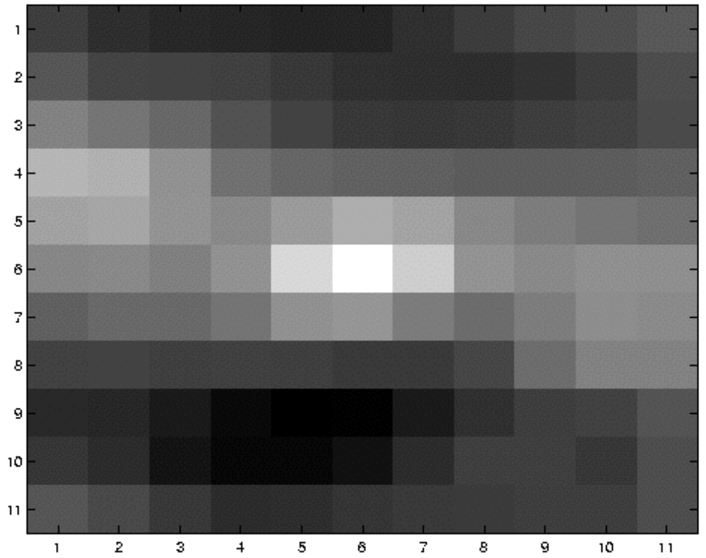
- Solution (pseudo-inverse):

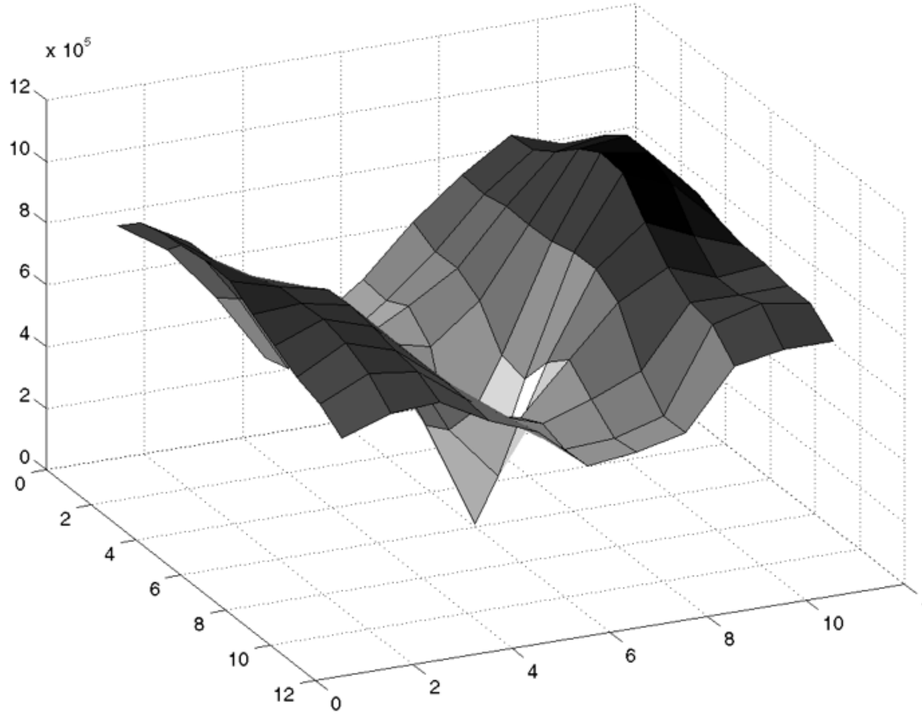
$$\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} A^T b$$

Think about it…

- Assuming the constant flow assumption is holding true, when does this method fail to give a solution?

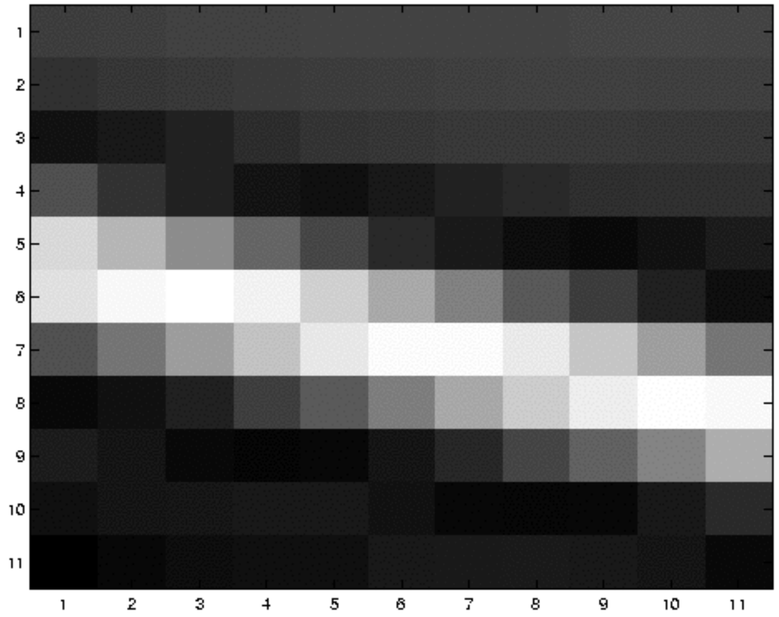
SSD Surface - Textured Area

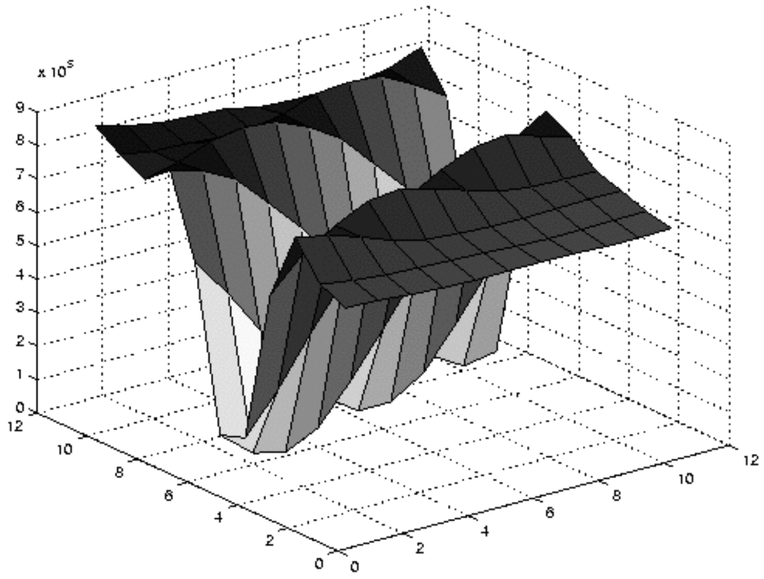




Slide from Szeliski/Seitz

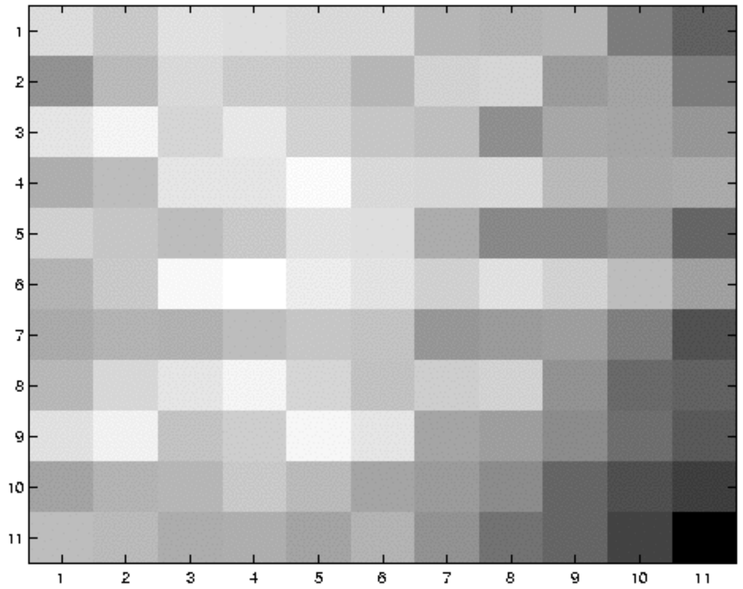
SSD Surface - Edge

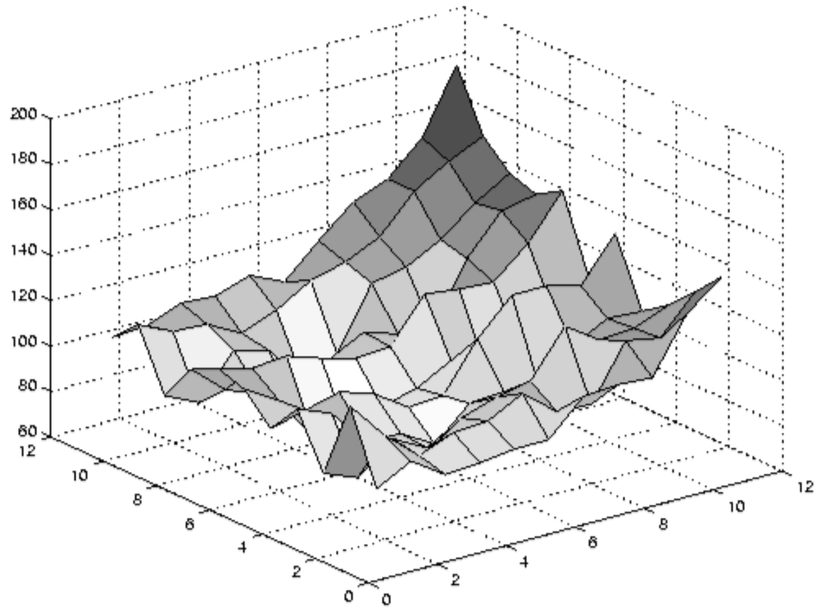




Slide from Szeliski/Seitz

SSD Surface - Homogeneous area





Slide from Szeliski/Seitz

Constant Flow Case

- $A^T A = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$
- Same as Harris matrix. Why?

Eigenvalue is near 0 =

Edge point

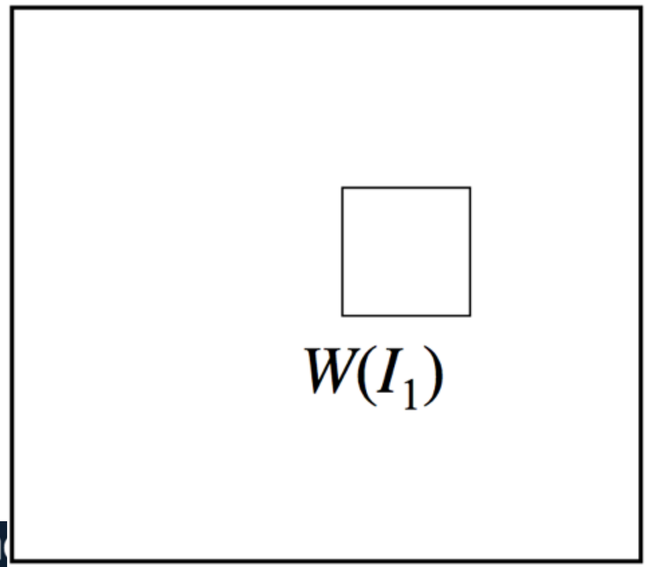
Applications

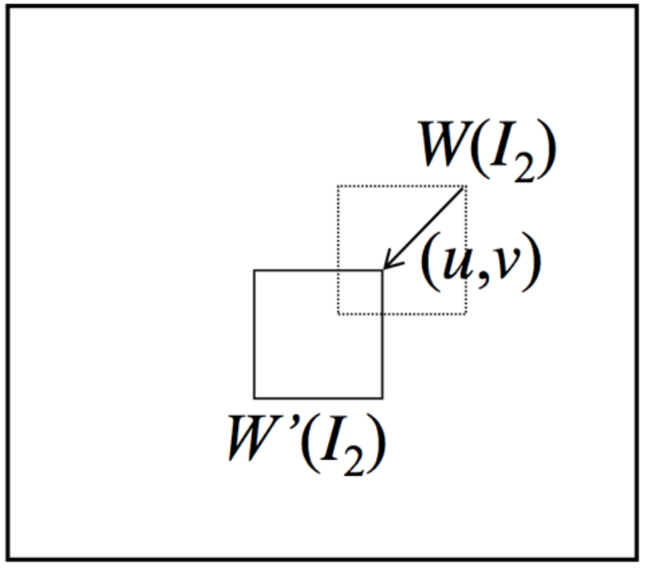
- Feature tracking:
  - Estimate (u,v) for one feature point across 2 frames
- Template tracking
  - Estimate the motion of a template in a video (e.g., track a face or a car through a video)
- Optical flow: (Later!!)
  - Estimate (u,v) at every pixel in (x,y)

- Given:
  - Images I_1 and I_2
  - Feature at (x_o,y_o) in I_1
  - Window W at (x_o,y_o)
- Compute displacement in image (u,v):

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- I_t is the pixel difference between $W(I_1)$ and $W(I_2)$
- Translate W by (u,v) to W'
- Compare $W(I_1)$ and $W'(I_2)$





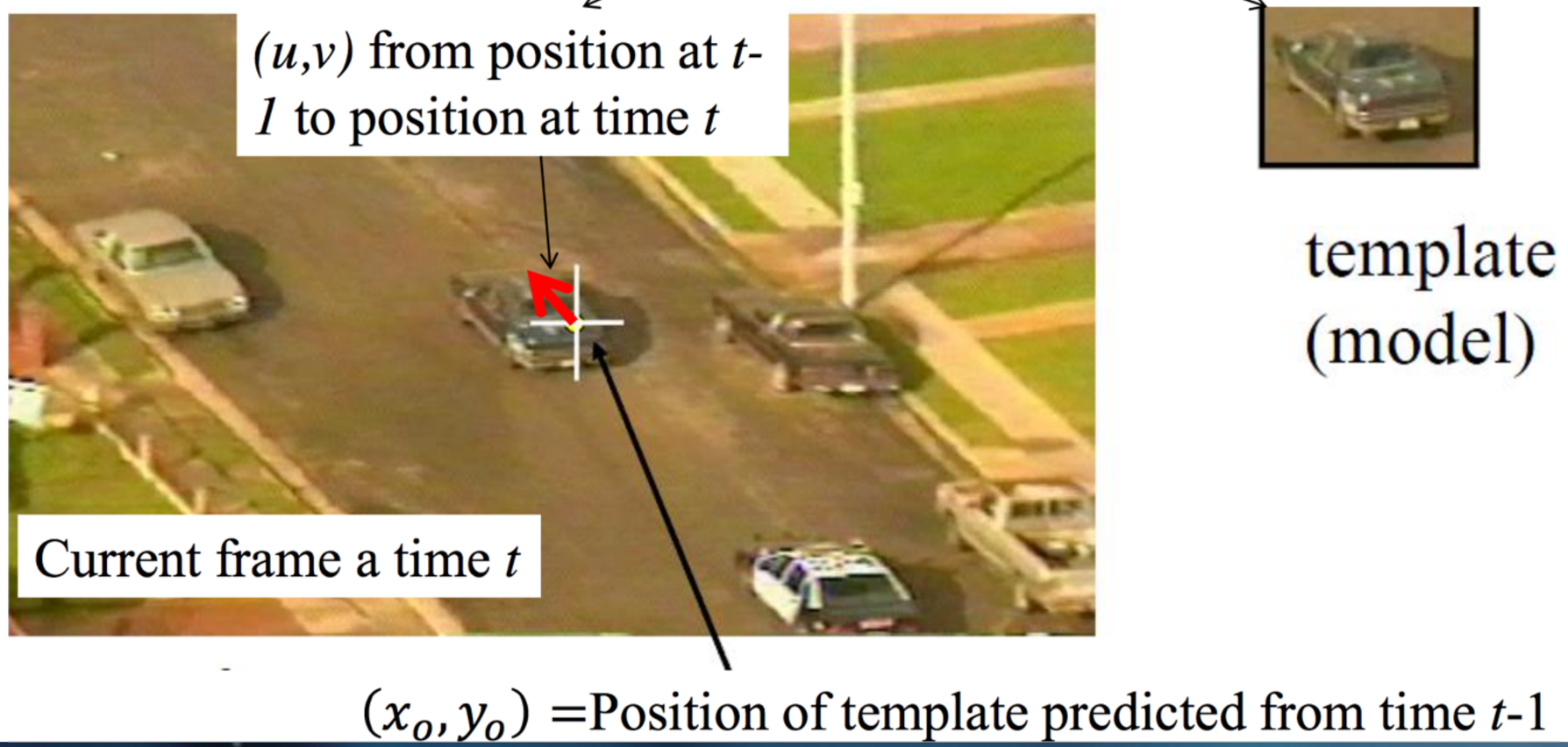
Feature Tracking (Lucas-Kanade)

- Single iteration estimates only subpixel motion
- Need to iterate (gradient descent)
- What if large (>> 1) motion:
  - Hierarchical tracking: Track at low resolution first and then increase resolution

Template Tracking

$$Min_{u,v} \sum_{W} \|I(x + x_o + u, y + y_o + v) - T(x, y)\|^2$$

- The same idea can be used to track a fixed template through a sequence of images. In that case, I_1 is a reference template T



Template Tracking : Method 1

- Exactly the same as for feature tracking, except with $I_t = I(x + x_o, y + y_o, t) - T(x, y)$ instead of $I_t = I(x, y, t) - I(x, y, t-1)$

- $\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} A^T b \qquad b = \begin{bmatrix} I_t \\ \vdots \\ I_t \end{bmatrix}$

- $A^T A = \begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_x I_y & \sum_W I_y^2 \end{bmatrix}$

- $I_t = I(x, y, t) - T(x - x_o, y - y_o)$

Template Tracking : Method 2

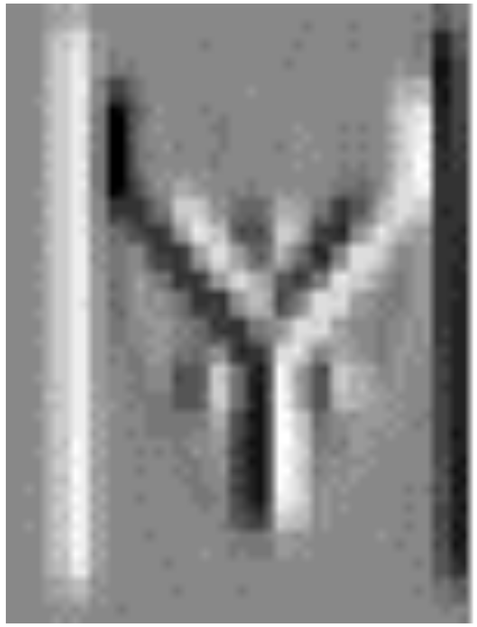
- Equivalently, we can solve instead:

$$Min_{u,v} \sum_{W} \|T(x-u, y-v) - I(x+x_o, y+y_o)\|$$

- $\begin{bmatrix} u \\ v \end{bmatrix} = -(A^T A)^{-1} A^T b \qquad b = \begin{bmatrix} I_t \\ \vdots \\ I_t \end{bmatrix}$
- The matrix $M = -(A^T A)^{-1} A^T$ is a 2xN matrix where N is the number of pixels in W
- $M = \begin{bmatrix} S_x^T \\ S_y^T \end{bmatrix}$ where S_x^T and S_y^T are each 1xN = the same size as W
- They depend only on the gradient of T → Can be pre-computed ***once***
- So $S_y^T b$ and $S_x^T b$ are simply the correlations between S_x and S_y and W
- Tracking = correlation with 2 images!!!

Template Tracking : Method 2

Template



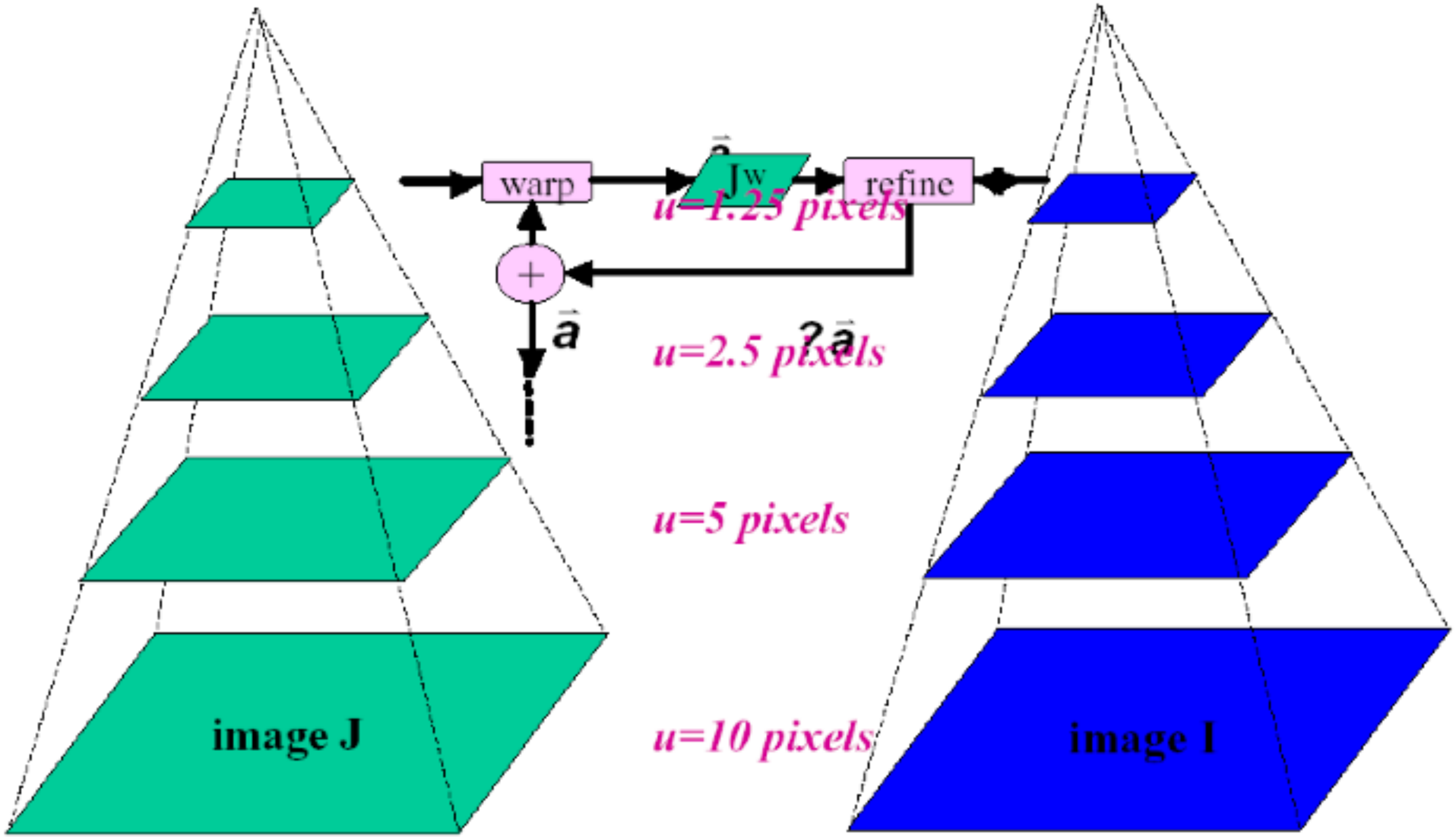
S_x

S_y

Issues

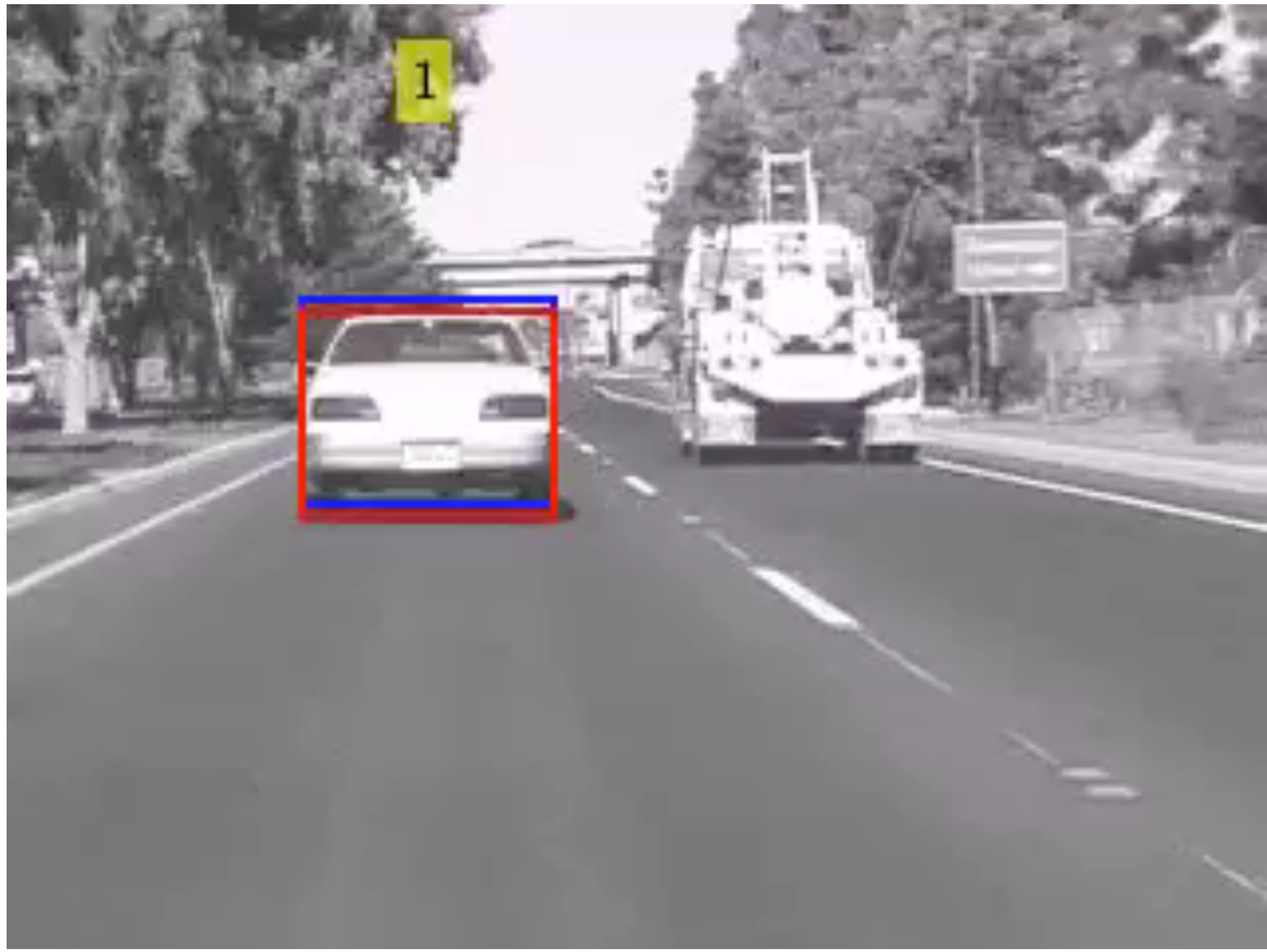
- Subpixel values→Multiple iterations (gradient descent)
- Dealing with large motions: Multiscale tracking
- Template drift
- Constant motion is too restrictive of an assumption, we need more general motion models→Affine quadratic (this lecture), mean shift (later in this class)
- Illumination and reflection models (end of this lecture)
- Dynamic models: We assumed that is the position from time t-1 but we could get a better estimate if we can guess the motion of the target→Motion models and data association (later in class)
- Occlusions: LKT requires the whole template to be visible→Robust tracking (later in class)

Dealing with Large Motions



Template drift

- Use fixed template: Lose track after large enough motion
- Update template every time (or often):
  - Small errors will accumulate over time
  - Confusion in case of occlusion
- Nasty problem, modern solution: Detection by tracking (later in class)



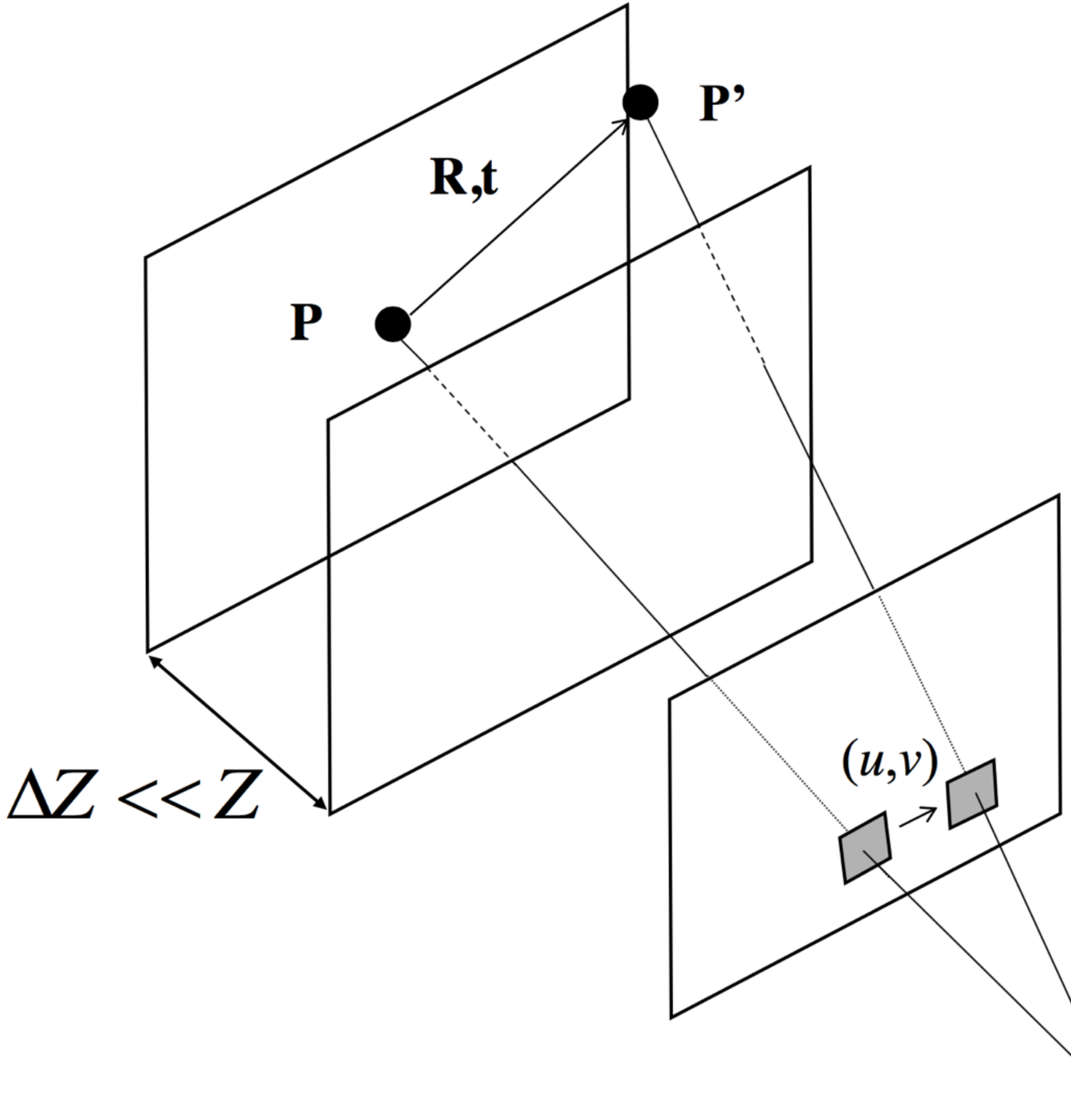
Limitations of Motion Models

- Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time.

- A more general model than u(x,y) = constant is needed

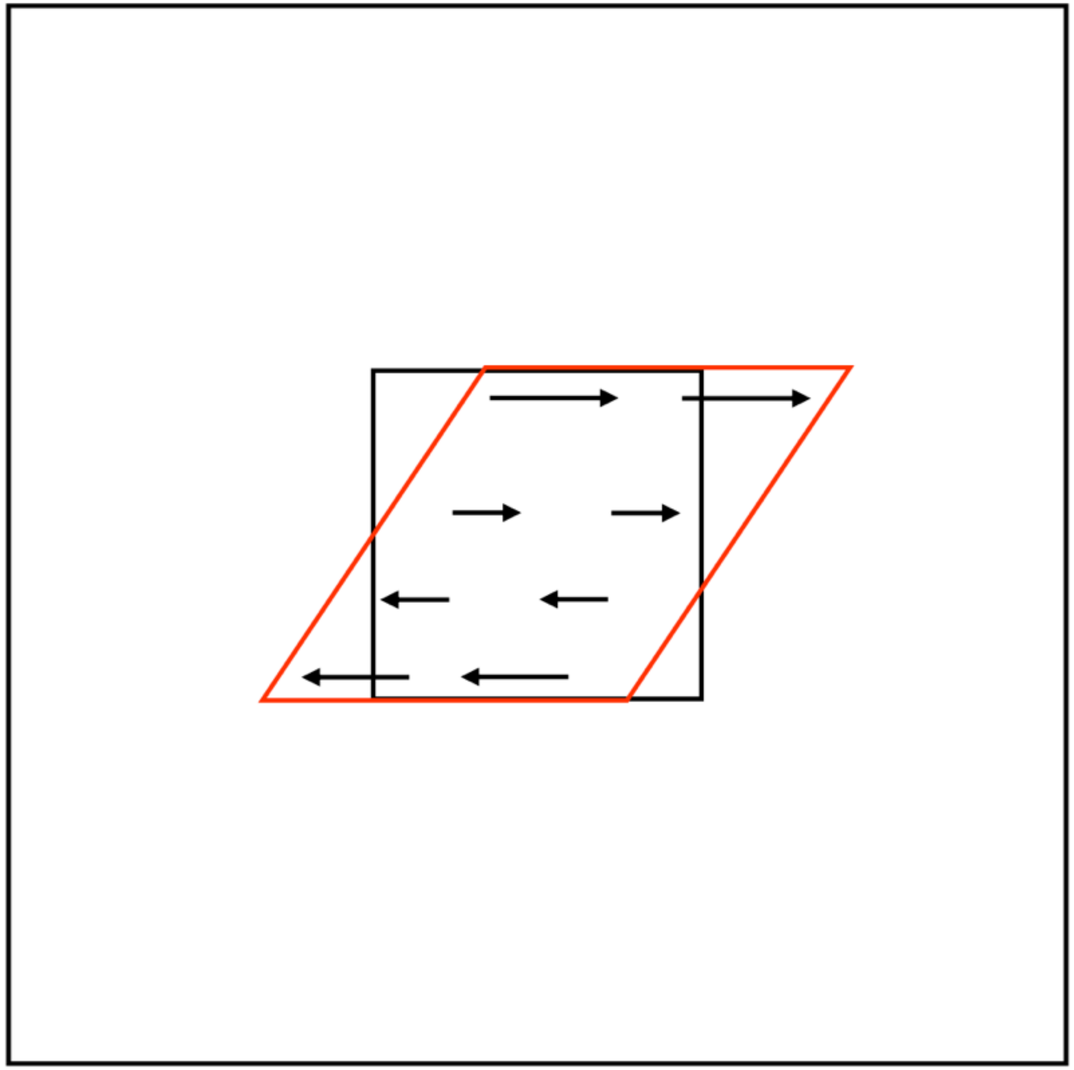
~~Constant Motion~~

DISTANT Motion!



Case 2 : Affine Motion

Affine Flow

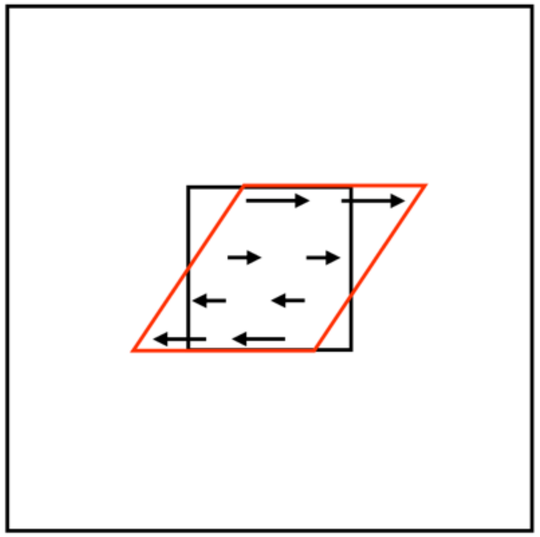


$$u(x,y) = ax + by + c$$

$$v(x,y) = dx + ey + f$$

Affine Flow

$$\underset{a,b,c,d,e,f}{\boldsymbol{Min}} \sum_{(x,y)\in W}\Big((ax+by+c)I_x+(dx+ey+f)I_y+I_t\Big)^2$$



$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = -\begin{bmatrix} \sum x^2I_x^2 & \sum xyI_x^2 & \sum xI_x^2 & \sum x^2I_xI_y & \sum xyI_xI_y & \sum xI_xI_y \\ & \sum y^2I_x^2 & \sum yI_x^2 & \sum xyI_xI_y & \sum y^2I_xI_y & \sum yI_xI_y \\ & & \sum I_x^2 & \sum xI_xI_y & \sum yI_xI_y & \sum I_xI_y \\ & & & \sum x^2I_y^2 & \sum xyI_y^2 & \sum xI_y^2 \\ & & & & \sum y^2I_y^2 & \sum yI_y^2 \\ & & & & & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum xI_xI_t \\ \sum yI_xI_t \\ \sum I_xI_t \\ \sum xI_yI_t \\ \sum yI_yI_t \\ \sum I_yI_t \end{bmatrix}$$

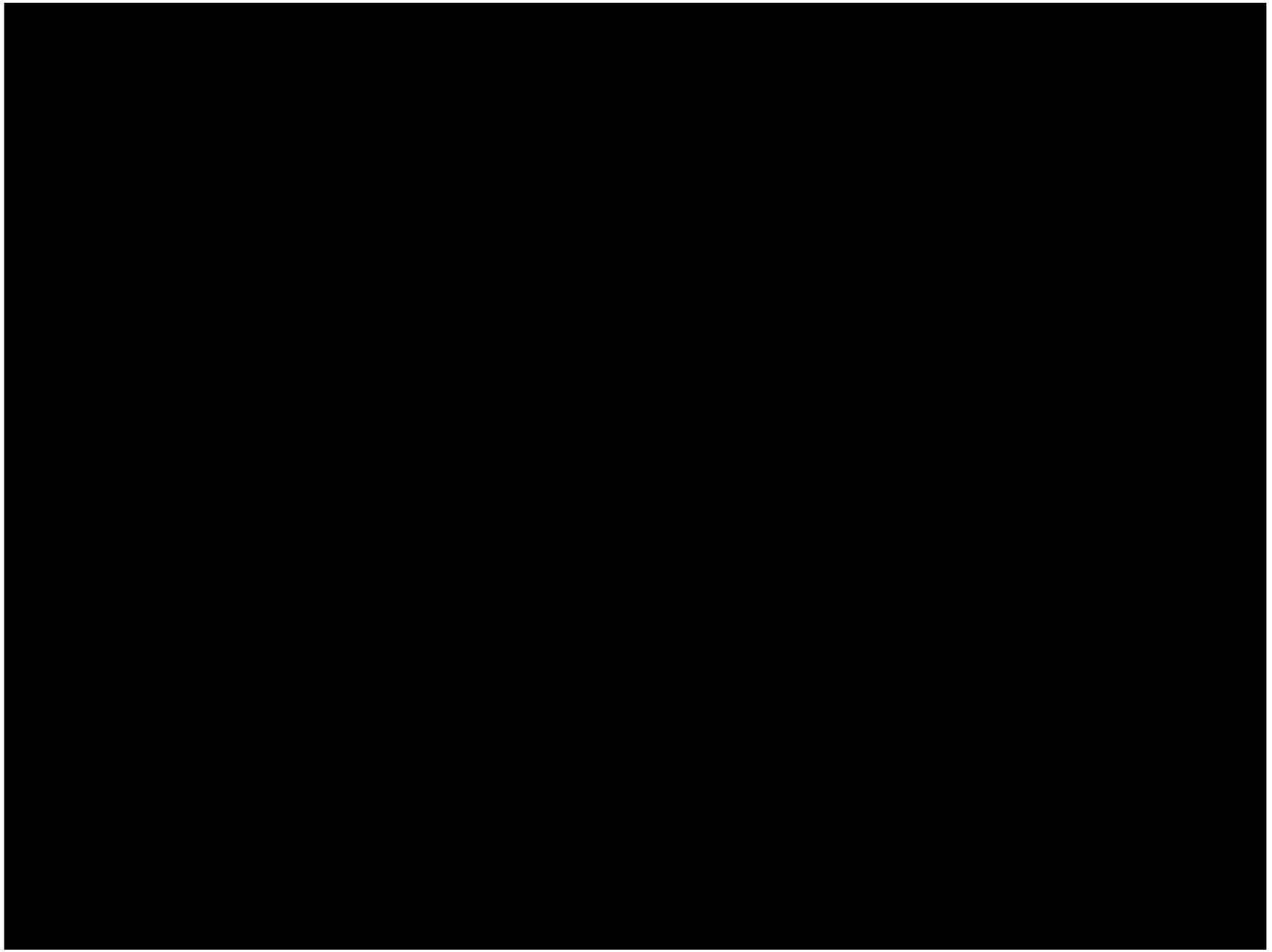




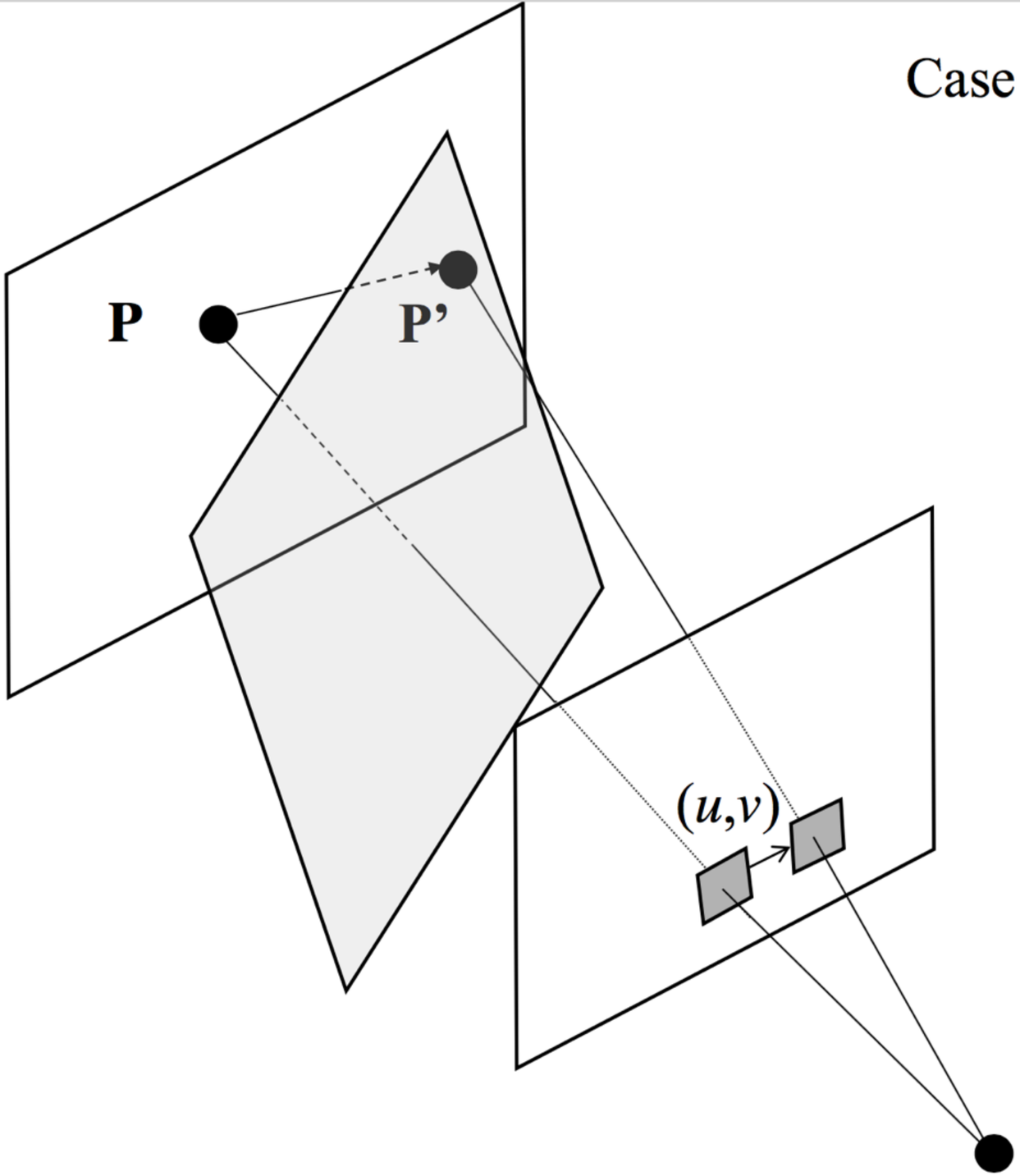

Tx Ty Rotation Scale Aspect Shear

- Still can be expressed as 6 correlations with 6 templates if tracking a fixed template
- Very fast
- Should do translation (= constant case) first

See also: **Lucas-Kanade 20 Years On: A Unifying Framework**
S. Baker and I. Matthews, *International Journal of Computer Vision*, Vol. 56, No. 3, March, 2004, pp. 221 - 255.



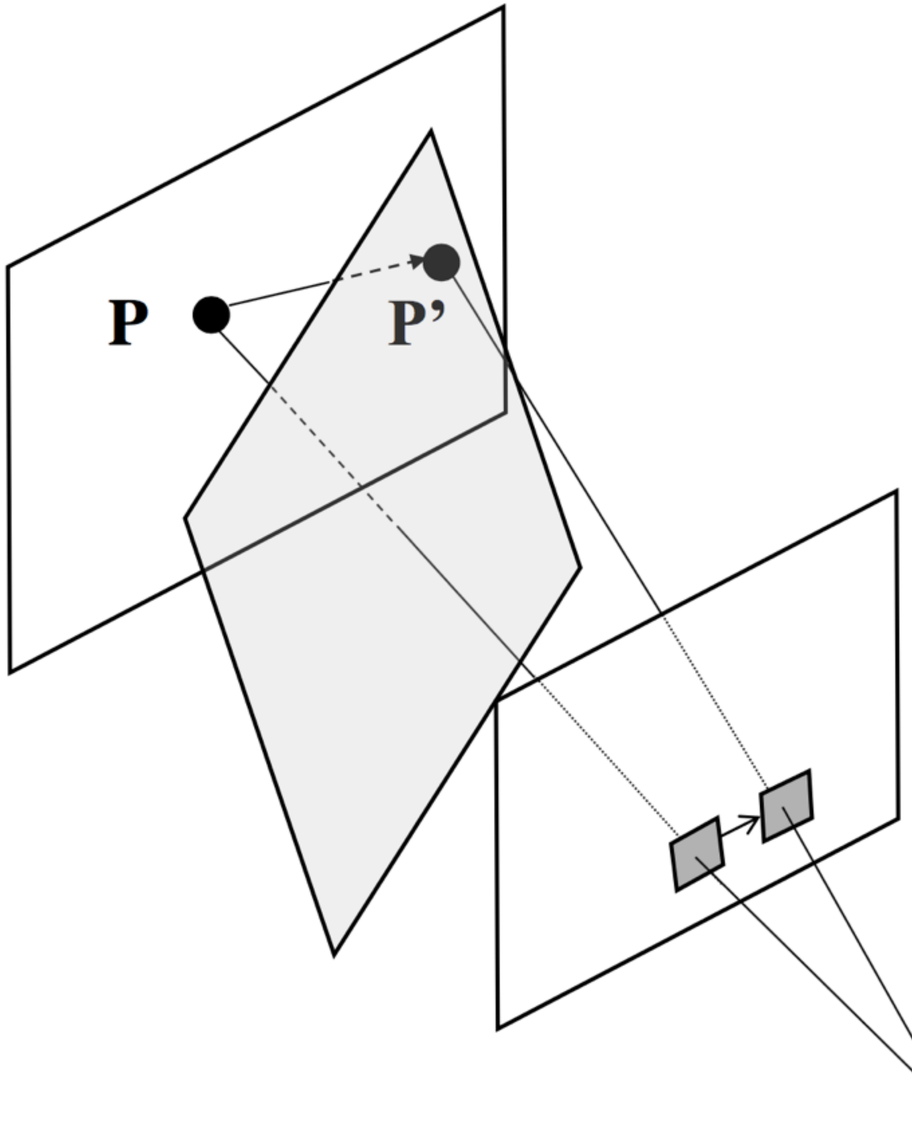
Case III: Scene is planar



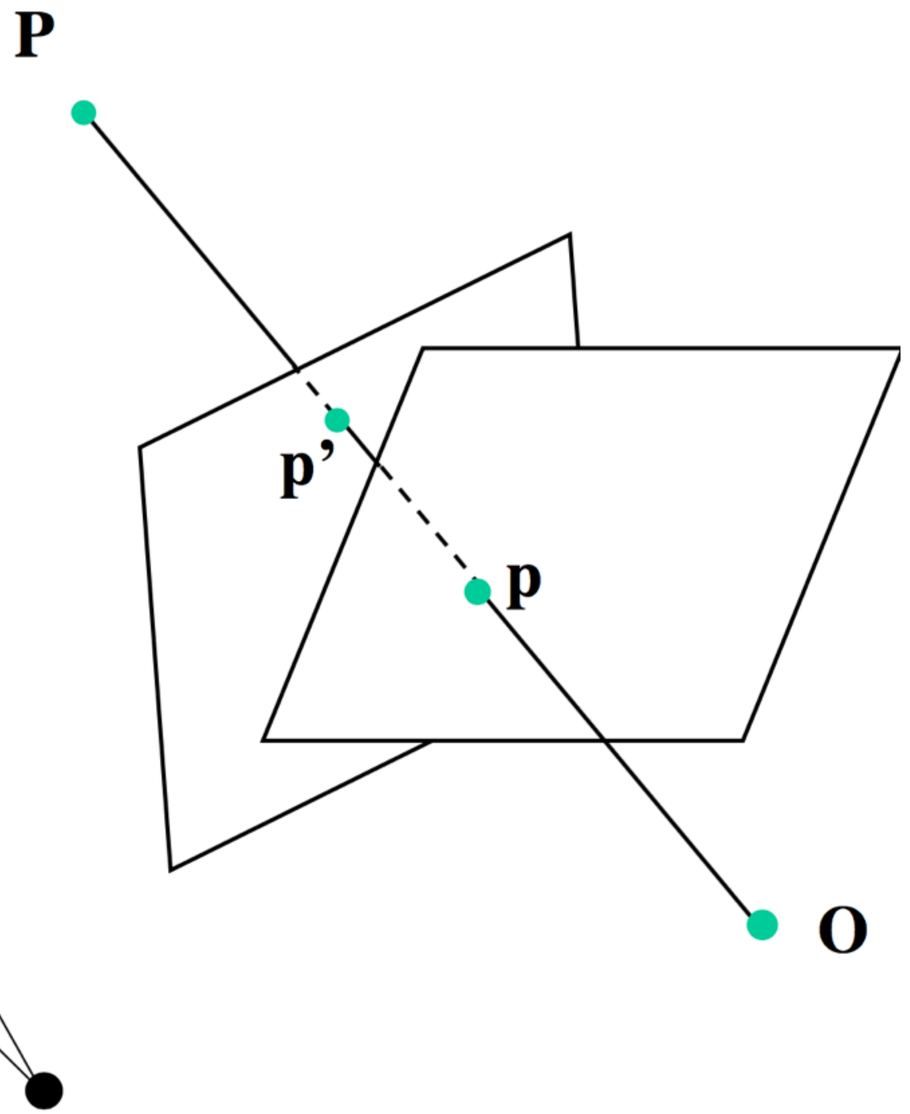
Think about it…

- From Homework 1 you can infer that this **homography** approach covers one more important case of camera motion
- What is it?

Scene motion



Camera rotation



$$\mathbf{p}' \equiv \mathbf{H}\mathbf{p}$$

Homographies

- We know how to solve this for planar scene or for pure rotation
- Can we take advantage of small motion between frames?