



# Lecture: Tracking

Juan Carlos Niebles and Ranjay Krishna  
Stanford Vision and Learning Lab



# CS 131 Roadmap

Tracking

21-Nov-2019





# What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>



# What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>



# Problem statement

## Image sequence



Slide credit: Yonsei Univ.



# Problem statement

## Feature point detection

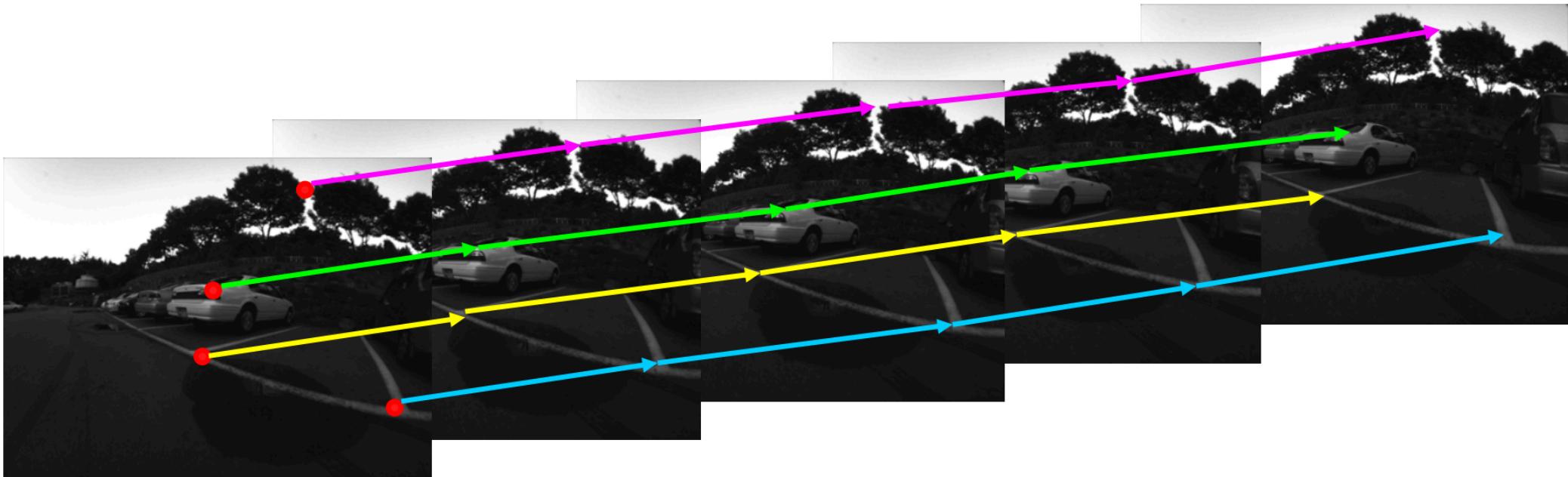


Slide credit: Yonsei Univ.



# Problem statement

## Feature point tracking



Slide credit: Yonsei Univ.



# Single object tracking





# Multiple object tracking





# Tracking with a fixed camera



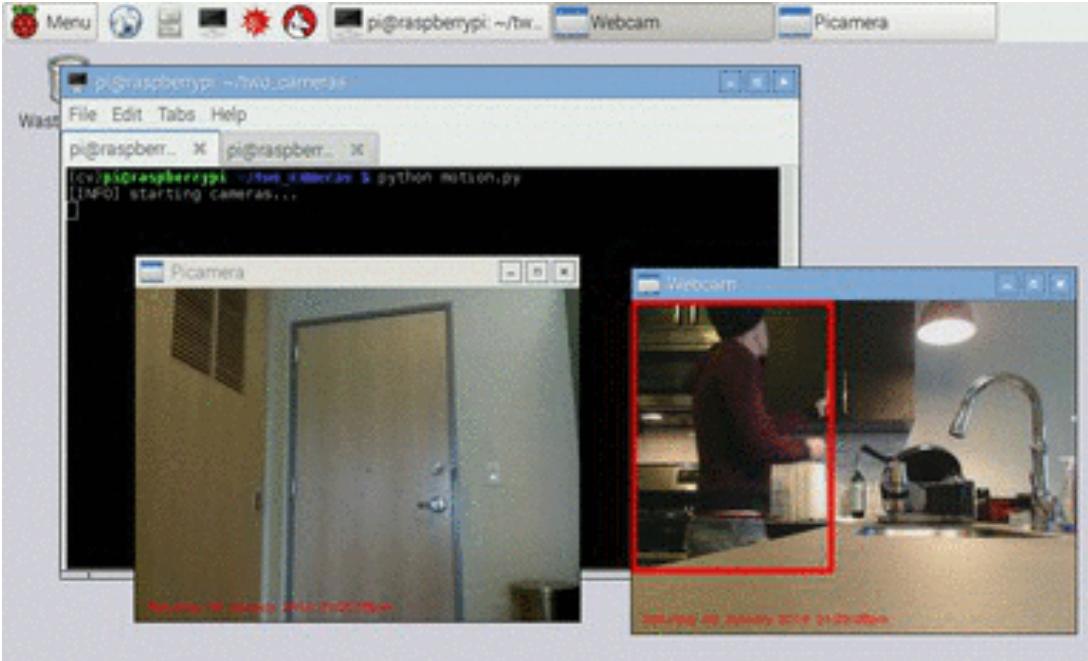


# Tracking with a moving camera





# Tracking with multiple cameras





# Challenges in Feature tracking

- Figure out which features can be tracked
  - Efficiently track across frames
- Some points may change appearance over time
  - e.g., due to rotation, moving into shadows, etc.
- Drift: small errors can accumulate as appearance model is updated
- Points may appear or disappear.
  - need to be able to add/delete tracked points.



# What are good features to track?

- Intuitively, we want to avoid smooth regions and edges. But is there a more principled way to define good features?
- **What kinds of image regions can we detect easily and consistently?** Think about what you learnt earlier in the class.



# What are good features to track?

- Can measure “quality” of features from just a single image.
- Hence: tracking Harris corners (or equivalent) guarantees small error sensitivity!

Source: Silvio Savarese



# Motion estimation techniques

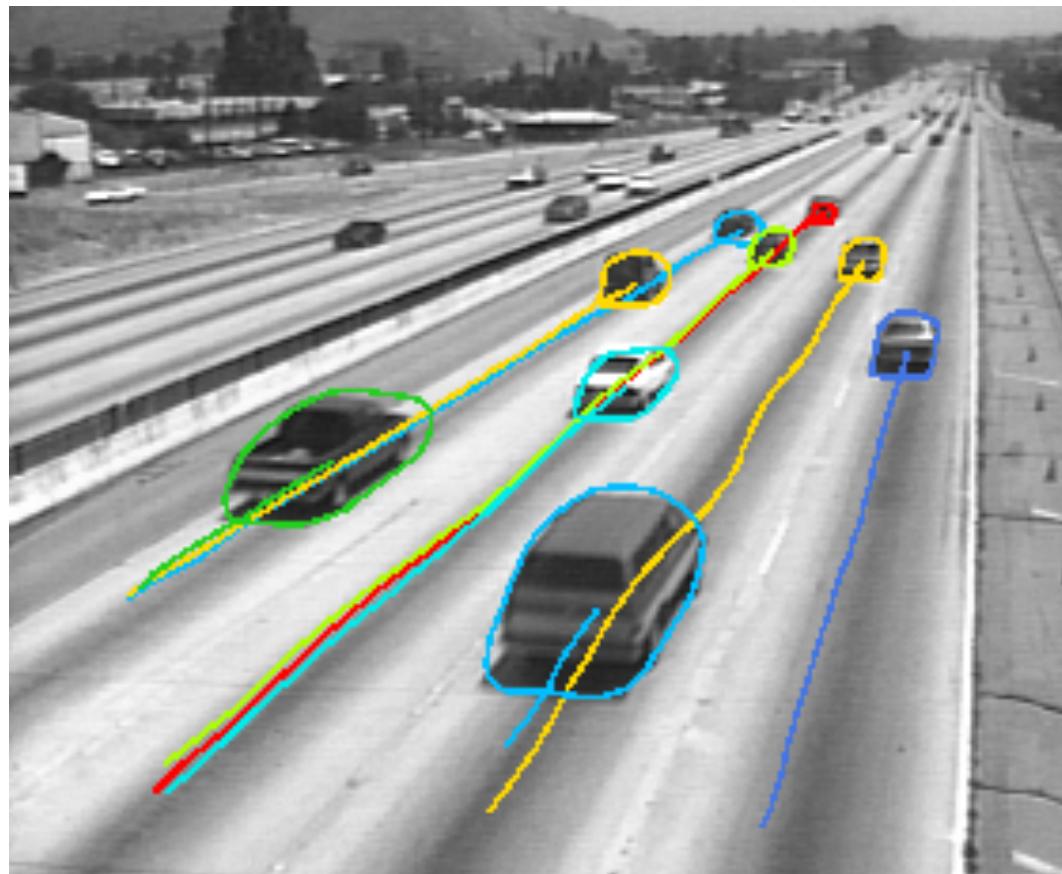
- Optical flow
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)
- Feature-tracking
  - Extract visual features (corners, textured areas) and “track” them over multiple frames





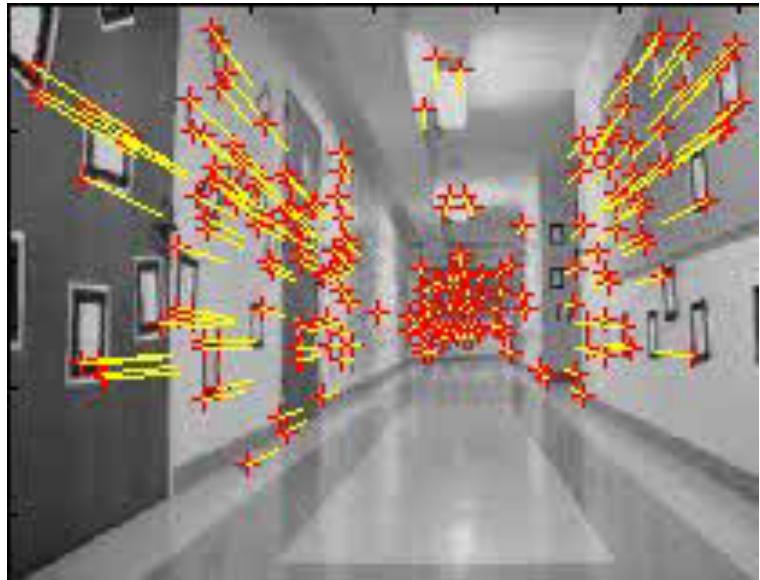
# Optical flow can help track features

Once we have the features we want to track, lucas-kanade or other optical flow algorithm can help track those features





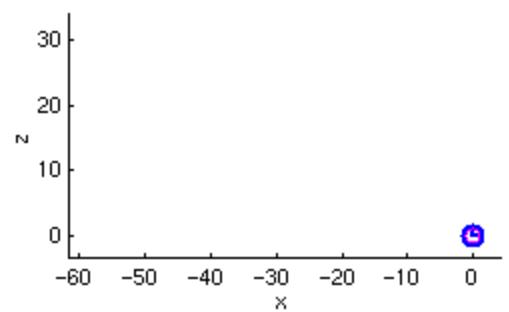
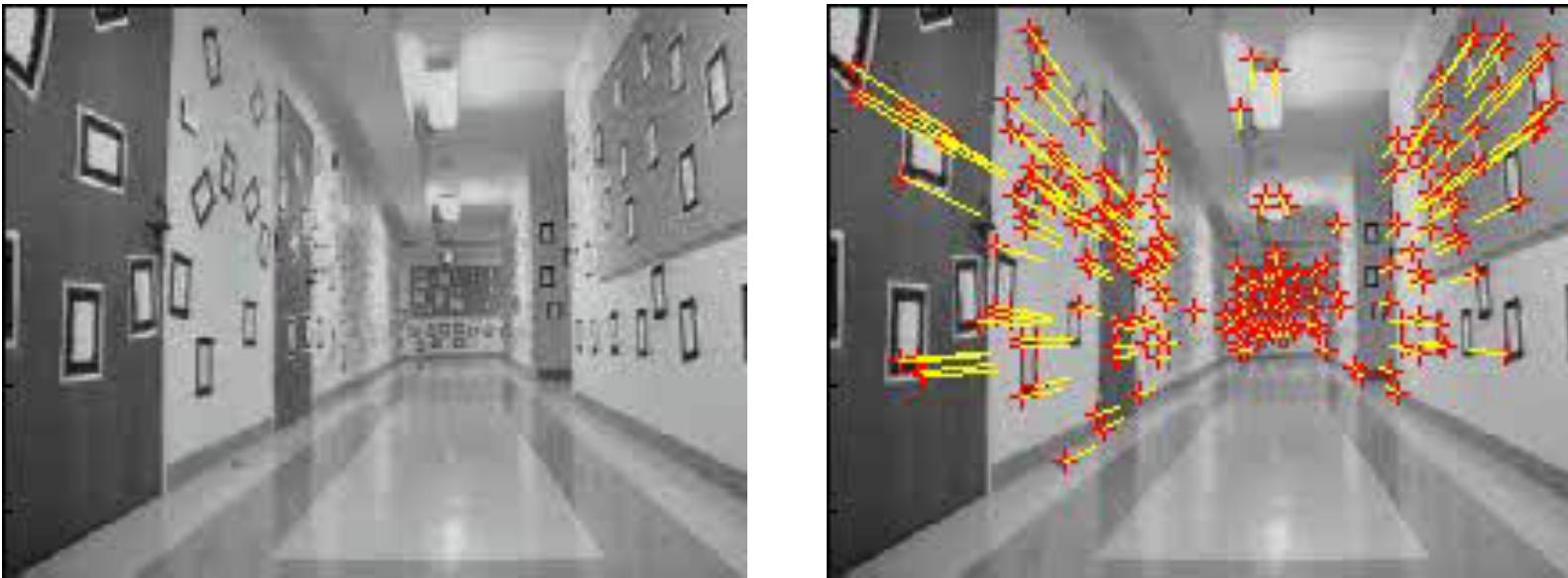
# Feature-tracking



Courtesy of Jean-Yves Bouguet – Vision Lab, California Institute of Technology



# Feature-tracking



Courtesy of Jean-Yves Bouguet – Vision Lab, California Institute of Technology



# What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>



# Simple KLT tracker

1. Find a good point to track (harris corner)
2. For each Harris corner compute motion (translation or affine) between consecutive frames.
3. Link motion vectors in successive frames to get a track for each Harris point
4. Introduce new Harris points by applying Harris detector at every m (10 or 15) frames
5. Track new and old Harris points using steps 1-3



# KLT tracker for fish



Video credit: Kanade



Tracking

21-Nov-2019

# Tracking cars



Video credit: Kanade

# Tracking movement



Video credit: Kanade



# What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

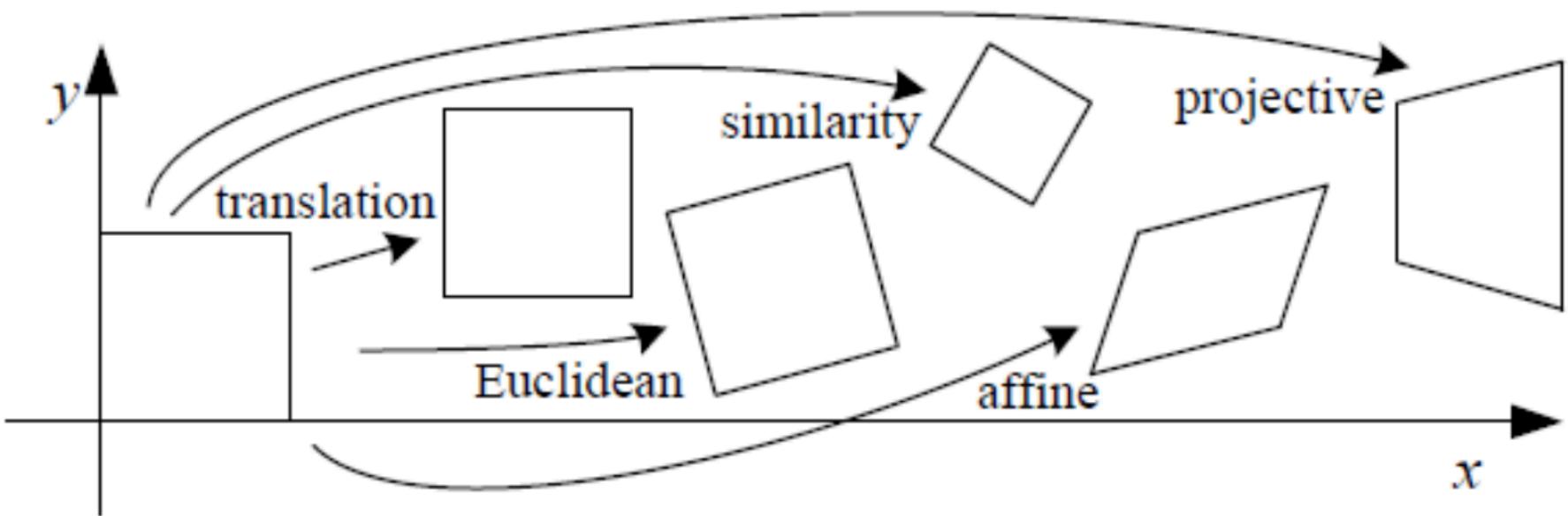
**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>

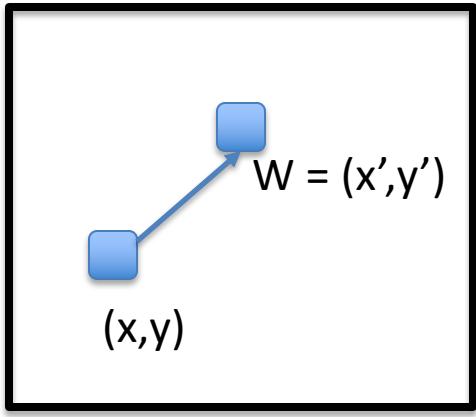


# Types of 2D transformations





# Translation

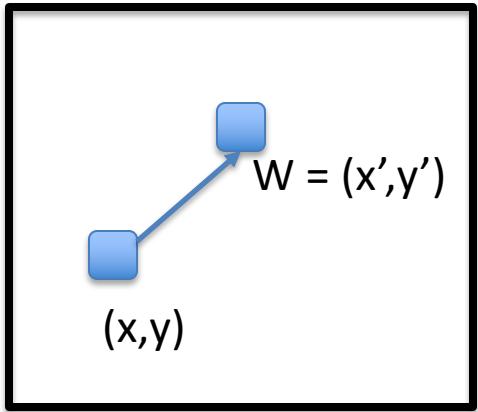


- Let the initial feature be located by  $(x, y)$ .
- In the next frame, it has translated to  $(x', y')$ .
- We can write the transformation as:

$$\begin{aligned}x' &= x + b_1 \\y' &= y + b_2\end{aligned}$$



# Translation



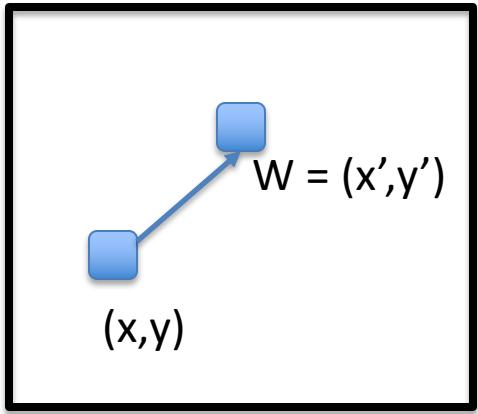
$$\begin{aligned}x' &= x + b_1 \\y' &= y + b_2\end{aligned}$$

- We can write this as a matrix transformation using homogeneous coordinates:

$$\bullet \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



# Translation



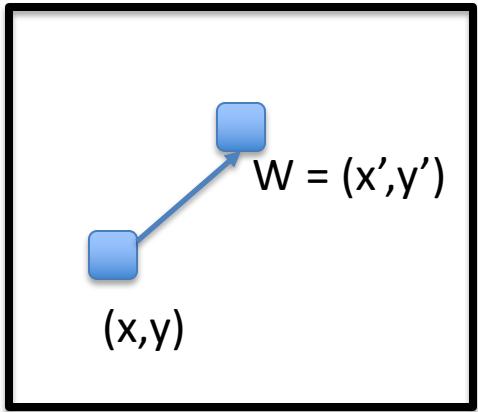
- $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Notation:

- $$W(x; p) = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



# Displacement Model for Translation



- $W(x; p) = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

There are only two parameters:

$$p = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The derivative of the transformation w.r.t.  $p$ :

$$\frac{\partial W}{\partial p}(x; p) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

This is called the Jacobian.



# Similarity motion

- Rigid motion includes scaling + translation.
- We can write the transformations as:

$$\begin{aligned}x' &= ax + b_1 \\y' &= ay + b_2\end{aligned}$$

- $W(x; p) = \begin{bmatrix} a & 0 & b_1 \\ 0 & a & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

- $p = [a \quad b_1 \quad b_2]^T$

- $\frac{\partial W}{\partial p}(x; p) = \begin{bmatrix} x & 1 & 0 \\ y & 0 & 1 \end{bmatrix}$



# Affine motion

- Affine motion includes scaling + rotation + translation.

$$x' = a_1x + a_2y + b_1$$

$$y' = a_3x + a_4y + b_2$$

$$\bullet W(x; p) = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\bullet p = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$$

$$\bullet \frac{\partial W}{\partial p}(x; p) = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$$



# What we will learn today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>



# Problem setting

- Given a video sequence, find all the features and track them across the video.
- First, use Harris corner detection to find features and their location  $x$ .
- For each feature at location  $x = [x \ y]^T$ :
  - Choose a descriptor create an initial template for that feature:  $T(x)$ .



# KLT objective

- Our aim is to find the  $\mathbf{p}$  that minimizes the difference between the template  $T(x)$  and the description of the new location of  $x$  after undergoing the transformation.

$$\sum_x [I(W(x; \mathbf{p})) - T(x)]^2$$

- For all the features  $x$  in the image  $I$ ,
  - $I(W(x; \mathbf{p}))$  is the estimate of where the features move to in the next frame after the transformation defined by  $W(x; \mathbf{p})$ . Recall that  $\mathbf{p}$  is our vector of parameters.
  - Sum is over an image patch around  $x$ .



# KLT objective

- Since  $\mathbf{p}$  may be large, minimizing this function may be difficult:

$$\sum_x [I(W(x; \mathbf{p})) - T(x)]^2$$

- We will instead break down  $\mathbf{p} = \mathbf{p}_0 + \Delta\mathbf{p}$ 
  - Large + small/residual motion
  - Where  $\mathbf{p}_0$  is going to be fixed and we will solve for  $\Delta\mathbf{p}$ , which is a small value.
  - We can initialize  $\mathbf{p}_0$  with our best guess of what the motion is and initialize  $\Delta\mathbf{p}$  as zero.



# A little bit of math: Taylor series

- Taylor series is defined as:

$$f(x + \Delta x) = f(x) + \Delta x \frac{\partial f}{\partial x} + \Delta x^2 \frac{\partial^2 f}{\partial x^2} + \dots$$

- Assuming that  $\Delta x$  is small.
- We can apply this expansion to the KLT tracker and only use the first two terms:



# Expanded KLT objective

$$\begin{aligned} & \sum_x [I(W(x; \mathbf{p}_0 + \Delta\mathbf{p})) - T(x)]^2 \\ & \approx \sum_x \left[ I(W(x; \mathbf{p}_0)) + \nabla I \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p} - T(x) \right]^2 \end{aligned}$$

It's a good thing we have already calculated what  $\frac{\partial W}{\partial \mathbf{p}}$  would look like for affine, translations and other transformations!



# Expanded KLT objective

- So our aim is to find the  $\Delta\mathbf{p}$  that minimizes the following:

$$\operatorname{argmin}_{\Delta\mathbf{p}} \sum_x \left[ I(W(\mathbf{x}; \mathbf{p}_0)) + \nabla I \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p} - T(x) \right]^2$$

- Where  $\nabla I = [I_x \quad I_y]$
- Differentiate wrt  $\Delta\mathbf{p}$  and setting it to zero:

$$\sum_x \left[ \nabla I \frac{\partial W}{\partial \mathbf{p}} \right]^T \left[ I(W(\mathbf{x}; \mathbf{p}_0)) + \nabla I \frac{\partial W}{\partial \mathbf{p}} \Delta\mathbf{p} - T(x) \right] = 0$$



# Solving for $\Delta p$

- Solving for  $\Delta p$  in:

$$\sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ I(W(x; p_0)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right] = 0$$

- we get:

$$\Delta p = H^{-1} \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T [T(x) - I(W(x; p_0))]$$

$$\text{where } H = \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ \nabla I \frac{\partial W}{\partial p} \right]$$



# Interpreting the H matrix for translation transformations

$$H = \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ \nabla I \frac{\partial W}{\partial p} \right]$$

Recall that

1.  $\nabla I = [I_x \quad I_y]$  and

2. for translation motion,  $\frac{\partial W}{\partial p}(x; p) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Therefore,

$$\begin{aligned} H &= \sum_x \left[ [I_x \quad I_y] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right]^T \left[ [I_x \quad I_y] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right] \\ &= \sum_x \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \end{aligned}$$

That's the Harris corner detector we learnt in class!!!



# Interpreting the H matrix for **affine** transformations

$$H = \sum_{\mathbf{x}} \begin{bmatrix} I_x^2 & I_x I_y & xI_x^2 & yI_x I_y & xI_x I_y & yI_x I_y \\ I_x I_y & I_y^2 & xI_x I_y & yI_y^2 & xI_y^2 & yI_y^2 \\ xI_x^2 & yI_x I_y & x^2 I_x^2 & y^2 I_x I_y & xyI_x I_y & y^2 I_x I_y \\ yI_x I_y & yI_y^2 & xyI_x I_y & y^2 I_y^2 & xyI_y^2 & y^2 I_y^2 \\ xI_x I_y & xI_y^2 & x^2 I_x I_y & xyI_y^2 & x^2 I_y^2 & xyI_y^2 \\ yI_x I_y & yI_y^2 & xyI_x I_y & y^2 I_y^2 & xyI_y^2 & y^2 I_y^2 \end{bmatrix}$$

Can you derive this yourself similarly to how we derived the translation transformation?



# Overall KLT tracker algorithm

$$\Delta \mathbf{p} = H^{-1} \sum_x \left[ \nabla I \frac{\partial W}{\partial \mathbf{p}} \right]^T [T(x) - I(W(x; \mathbf{p}_0))]$$

Given the features from Harris detector:

1. Initialize  $\mathbf{p}_0$  and  $\Delta \mathbf{p}$ .
2. Compute the initial templates  $T(x)$  for each feature.
3. Transform the features in the image  $I$  with  $W(x; \mathbf{p}_0)$ .
4. Measure the error:  $I(W(x; \mathbf{p}_0)) - T(x)$ .
5. Compute the image gradients  $\nabla I = [I_x \quad I_y]$ .
6. Evaluate the Jacobian  $\frac{\partial W}{\partial \mathbf{p}}$ .
7. Compute steepest descent  $\nabla I \frac{\partial W}{\partial \mathbf{p}}$ .
8. Compute Inverse Hessian  $H^{-1}$
9. Calculate the change in parameters  $\Delta \mathbf{p}$
10. Update parameters  $\mathbf{p}_0 = \mathbf{p}_0 + \Delta \mathbf{p}$
11. Repeat 2 to 10 until  $\Delta \mathbf{p}$  is small.



# KLT over multiple frames

- Once you find a transformation for two frames, you will repeat this process for every couple of frames.
- Run Harris detector every 15-20 frames to find new features.



# Challenges to consider

- Implementation issues
- Window size
  - Small window more sensitive to noise and may miss larger motions (without pyramid)
  - Large window more likely to cross an occlusion boundary (and it's slower)
  - 15x15 to 31x31 seems typical
- Weighting the window
  - Common to apply weights so that center matters more (e.g., with Gaussian)



# What we learnt today?

- Feature Tracking
- Simple KLT tracker
- 2D transformations
- Iterative KLT tracker

**Reading:** [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]

<http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf>