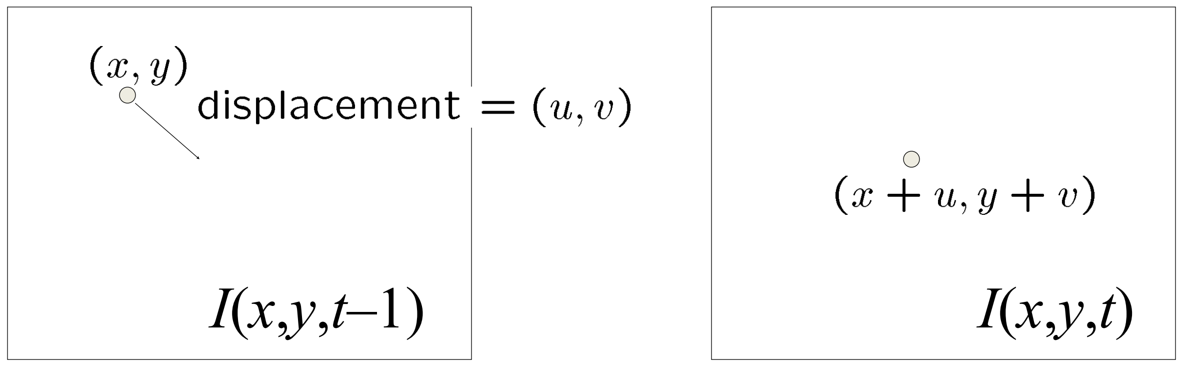
CSE 152: Computer Vision

Hao Su

Lecture 16: Tracking

Review of Lucas-Kanade Algorithm



- Brightness Constancy Equation:

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

Linearizing the right side using Taylor expansion:

Image derivative along x

$$I(x+u, y+v, t) \approx I(x, y, t-1) + \boxed{I_x} \cdot u(x, y) + I_y \cdot v(x, y) + \boxed{I_t}$$

$$I(x+u, y+v, t) - I(x, y, t-1) = I_x \cdot u(x, y) + I_y \cdot v(x, y) + I_t$$

Hence, $I_x \cdot u + I_y \cdot v + I_t \approx 0 \quad \rightarrow \nabla I \cdot [u \; v]^T + I_t = 0$

Source: Silvio Savarese

Ambiguity of estimation

Can we use this equation to recover image motion (u,v) at each pixel?

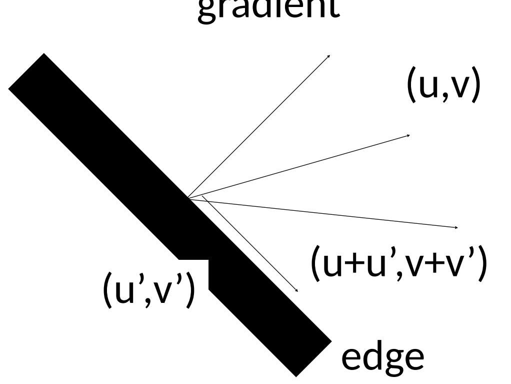
$$\nabla I \cdot [u\ v]^T + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

The component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation,
so does (u+u', v+v') if

$$\nabla I \cdot [u'\ v']^T = 0$$



Source: Silvio Savarese

Solving the ambiguity…

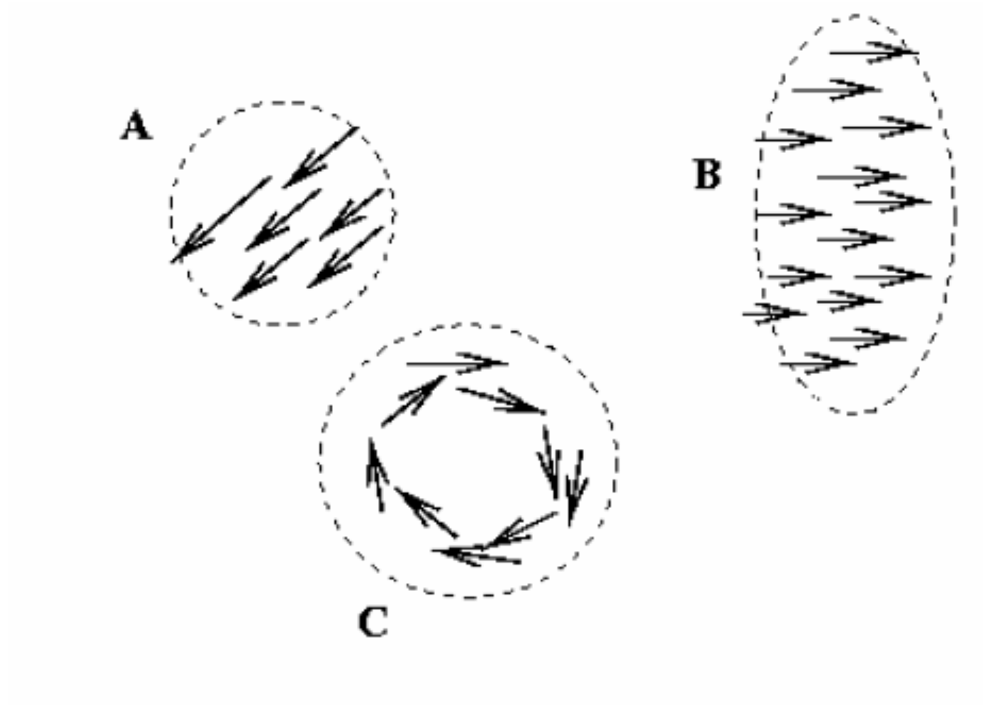
B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

- How to get more equations for a pixel?
- Spatial coherence constraint:
- Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

Source: Silvio Savarese

Problem of this solution



- Good for case A and case B, but not case C
- Underlying assumption (spatial coherency) of the stated method:
  - local neighborhood shares the same **translation**

Can we relax the translation coherency assumption for more general motions?

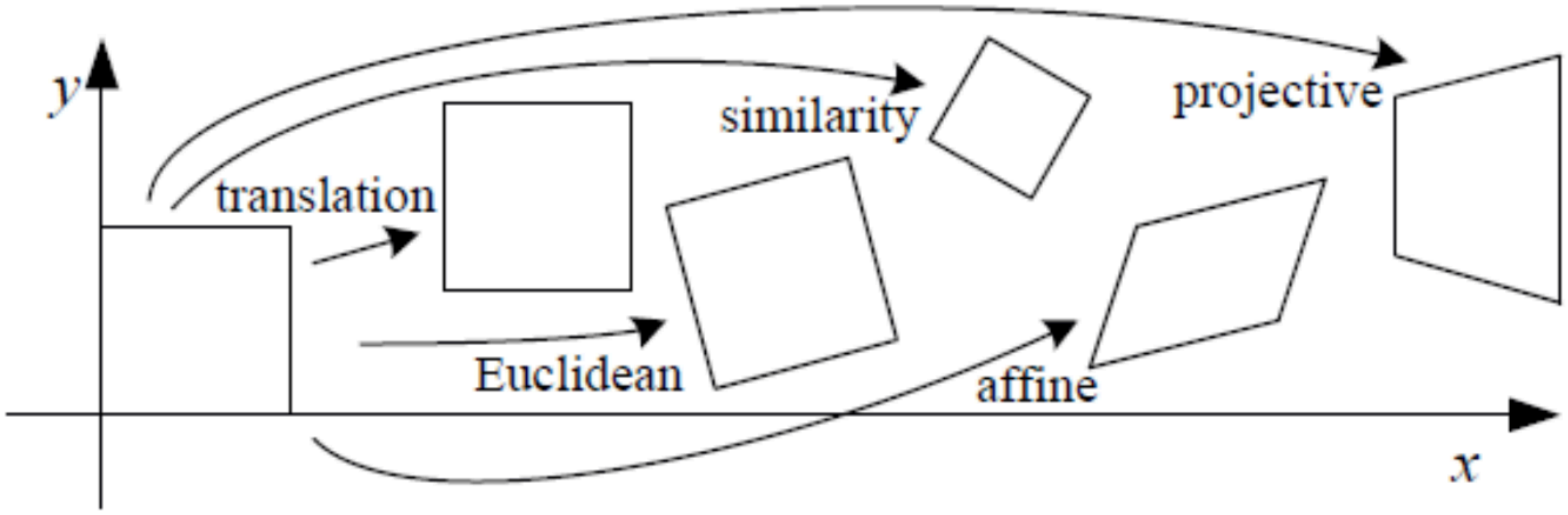
What we will learn today?

- 2D transformations
- Iterative KLT tracker

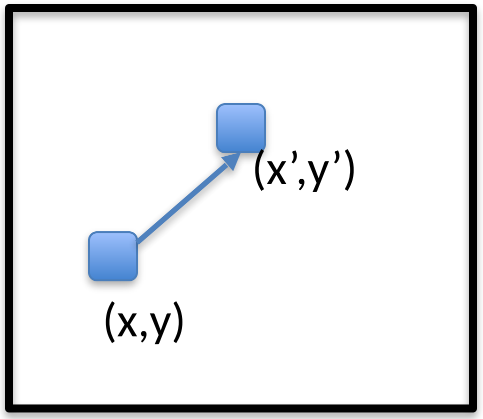
Reading: [Szeliski] Chapters: 8.4, 8.5

[Fleet & Weiss, 2005]
http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf

Types of 2D transformations

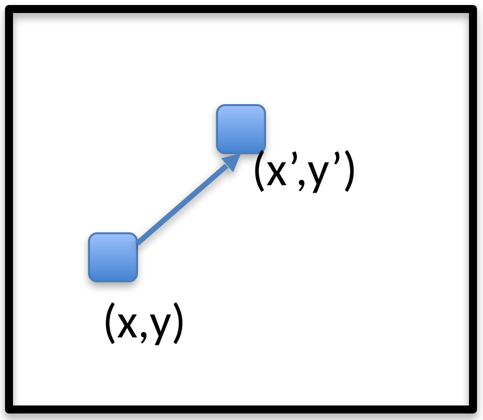


Translation



- 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:
  $x' = x + b_1$
  $y' = y + b_2$

Translation

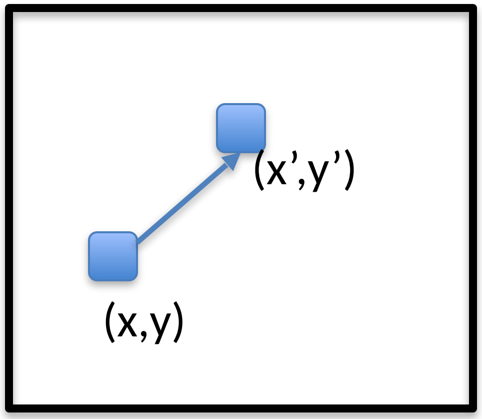


- $x' = x + b_1$
  $y' = y + b_2$

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

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

Translation

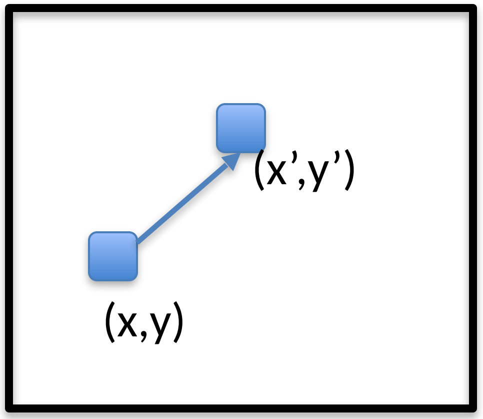


- $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- We will write the above transformation:

$$P = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{bmatrix}$$

Displacement Model for Translation



$$W(\boldsymbol{x};\theta) = \begin{bmatrix} 1 & 0 & b1 \\ 0 & 1 & b2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

There are only two parameters:

$$\theta = \begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

The derivative of the transformation w.r.t. θ:

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

- This is called the Jacobian.

Similarity motion

- Rigid motion includes scaling + translation.
- We can write the transformations as:
  $x' = ax + b_1$
  $y' = ay + b_2$
- $P = \begin{bmatrix} a & 0 & b_1 \\ 0 & a & b_2 \end{bmatrix}$
- $\theta = [a \quad b_1 \quad b_2]^T$
- $\frac{\partial W}{\partial \theta}(\boldsymbol{x};\theta) = \begin{bmatrix} x & 1 & 0 \\ y & 0 & 1 \end{bmatrix}$

Affine motion

- Affine motion includes scaling + rotation + translation.
- $x' = a_1x + a_2y + b_1$

  $y' = a_3x + a_4y + b_2$
- $P = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{bmatrix}$
- $\theta = [a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$
- $\dfrac{\partial W}{\partial \theta}(\boldsymbol{x};\theta) = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix}$

What we will learn today?

- 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 formulation

- Given a video sequence, find all the features and track them across the video.
- First, use Harris corner detection to find the feature points.
- For each feature at location $\mathbf{x} = [x \;\; y]^T$:
  - Choose a feature descriptor, e.g., SIFT, to create an initial template for that feature: $T(\boldsymbol{x})$.
  - The feature descriptor can also be the vanilla pixel intensity or RGB value: $I(x)$

KLT objective

- Our aim is to minimize the difference between the template T(x) and the description of the new location of x after undergoing the transformation.

$$\sum_{x} [T(W(x;\theta)) - T(x)]^2$$

- For all the feature points $\boldsymbol{x}$ in the image $\boldsymbol{I}$,
  - $T(W(x;\theta))$ is the feature of where the pixel move to in the next frame after the transformation defined by $W(x;\theta)$.

KLT objective

- Instead of minimizing this function:

$$\sum_x [T(W(x;\theta)) - T(x)]^2$$

- We will represent $\theta = \theta_0 + \Delta\theta$
  - Where θ_0 is going to be fixed and we will solve for $\Delta\theta$, which is a small value.
- We can initialize θ_0 with our best guess of what the motion is and initialize $\Delta\theta$ as zero.

A little bit of math: Taylor series

- Taylor series is defined as:

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

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

Expanded KLT objective

$$\sum_x [T(W(x;\theta_0 + \Delta\theta)) - T(x)]^2$$

$$\approx \sum_x \left[T(W(x;\theta_0)) + \nabla T \left.\frac{\partial W}{\partial \theta}\right|_{\theta_0} \Delta\theta - T(x)\right]^2$$

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

Expanded KLT objective

- So our aim is to find the $\Delta\theta$ that minimizes the following:

$$\underset{\Delta\theta}{\operatorname{argmin}} \sum_{x} [T(W(x;\theta_0)) + \nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \Delta\theta - T(x)]^2$$

Where $\nabla T = [\frac{\partial T}{\partial x} \quad \frac{\partial T}{\partial y}]$

- Differentiate w.r.t $\Delta\theta$ and setting it to zero:

$$\sum_{x} \left[\nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \right]^T \left[T(W(x;\theta_0)) + \nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \Delta\theta - T(x) \right] = 0$$

Solving for $\Delta\theta$

- Solving for $\Delta\theta$ in:

$$\sum_x \left[\nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \right]^T \left[T(W(x;\theta_0)) + \nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \Delta\theta - T(x) \right] = 0$$

- we get:

$$\Delta\theta = H^{-1} \sum_x \left[\nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \right]^T [T(x) - T(W(x;\theta_0))]$$

where $H = \sum_x \left[\nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \right]^T \left[\nabla T \frac{\partial W}{\partial \theta}\bigg|_{\theta_0} \right]$

Interpreting the H matrix for translation transformations

$$H = \sum_x \left[\nabla T \frac{\partial W}{\partial \theta}\Big|_{\theta_0} \right]^T \left[\nabla T \frac{\partial W}{\partial \theta}\Big|_{\theta_0} \right]$$

Suppose $T = I$, i.e., we use pixel intensive as the feature. Recall that

1. $\nabla I = [I_x \quad I_y]$ and
2. for translation motion, $\frac{\partial W}{\partial \theta}(x;\theta) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Therefore,

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

$$= \boxed{\sum_x \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}$$

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

Interpreting the H matrix for affine transformations

$$H = \sum_{\mathbf{x}} \begin{bmatrix} I_x^2 & I_x I_y & x I_x^2 & y I_x I_y & x I_x I_y & y I_x I_y \\ I_x I_y & I_y^2 & x I_x I_y & y I_y^2 & x I_y^2 & y I_y^2 \\ x I_x^2 & y I_x I_y & x^2 I_x^2 & y^2 I_x I_y & xy I_x I_y & y^2 I_x I_y \\ y I_x I_y & y I_y^2 & xy I_x I_y & y^2 I_y^2 & xy I_y^2 & y^2 I_y^2 \\ x I_x I_y & x I_y^2 & x^2 I_x I_y & xy I_y^2 & x^2 I_y^2 & xy I_y^2 \\ y I_x I_y & y I_y^2 & xy I_x I_y & y^2 I_y^2 & xy I_y^2 & y^2 I_y^2 \end{bmatrix}$$

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

Overall KLT tracker algorithm

Given the features from Harris detector:

1. Initialize θ_0 and $\Delta\theta$.
2. Compute the initial templates $T(x)$ for each feature.
3. Transform the features in the image I with $W(x;\theta_0)$.
4. Measure the error: $T(W(x;\theta_0)) - T(x)$.
5. Compute the image gradients $\nabla T = [T_x \quad T_y]$.
6. Evaluate the Jacobian $\left.\frac{\partial W}{\partial \theta}\right|_{\theta_0}$.
7. Compute steepest descent $\nabla T \left.\frac{\partial W}{\partial \theta}\right|_{\theta_0}$.
8. Compute Inverse Hessian H^{-1}
9. Calculate the change in parameters $\Delta\theta$
10. Update parameters $\theta = \theta_0 + \Delta\theta$

Iterative KLT

- 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)