Lecture 14.
Video Tracking

Pattern Recognition and Computer Vision

Guanbin Li,
School of Computer Science and Engineering, Sun Yat-Sen University

扫码签到

What we will learn today?

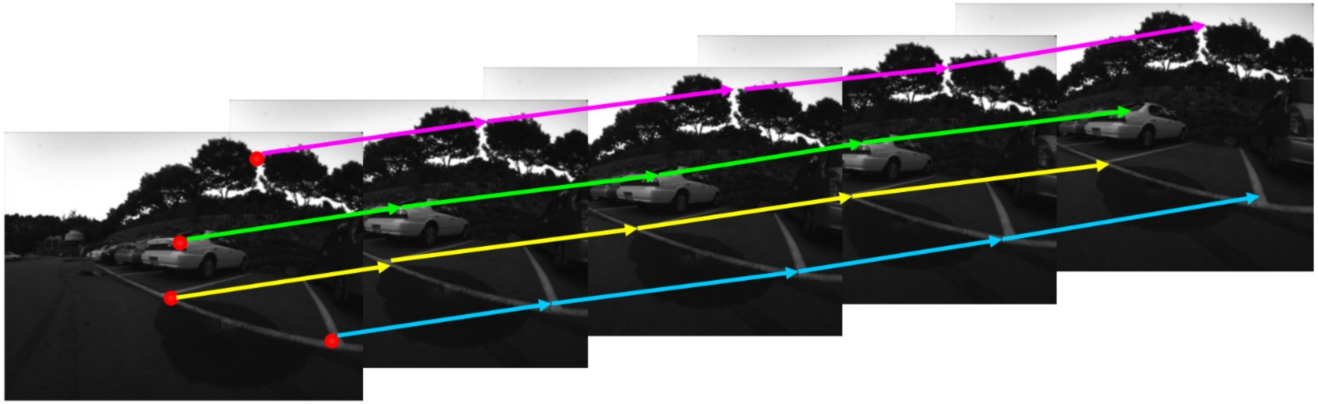
- Video Tracking
- Feature Tracking
- Simple KLT tracker
- Higher-Order Motion Model

Video Tracking

- Problem statement
  - Given a video $V = \{F_1, F_2, \dots, F_N\}$, which consists of N frames.
  - **Our goal** is track *features* or *objects* of interest in each frame F_i.

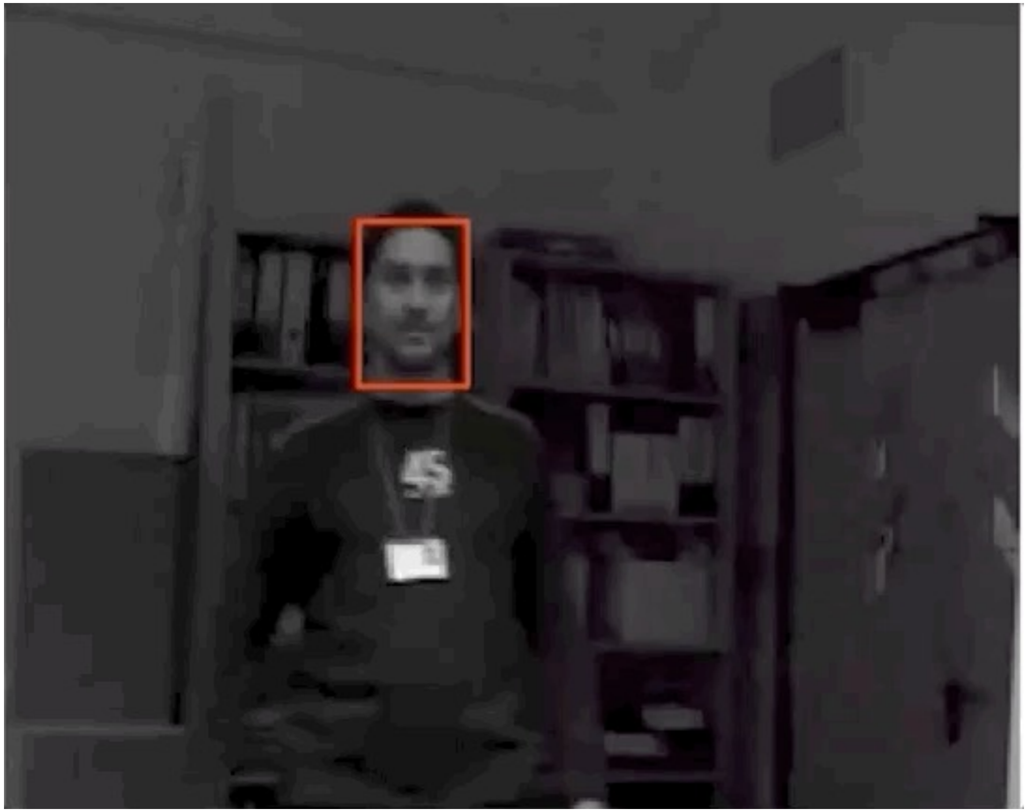
Video Tracking

- Feature point tracking



Video Tracking

- Single object tracking



Video Tracking

- Multiple object tracking

Video Tracking

- Why do video tracking?
  - Why not directly detect features or objects in each frame?
  - Here are some reasons:
    - tracking allows to maintain object identities
    - detection is computationally expensive
    - tackle challenging common problems
      - change of illumination or scale
      - motion blur
      - occlusions
      - poor quality of the image

What we will learn today?

- Video Tracking
- Feature Tracking
- Simple KLT tracker
- Higher-Order Motion Model

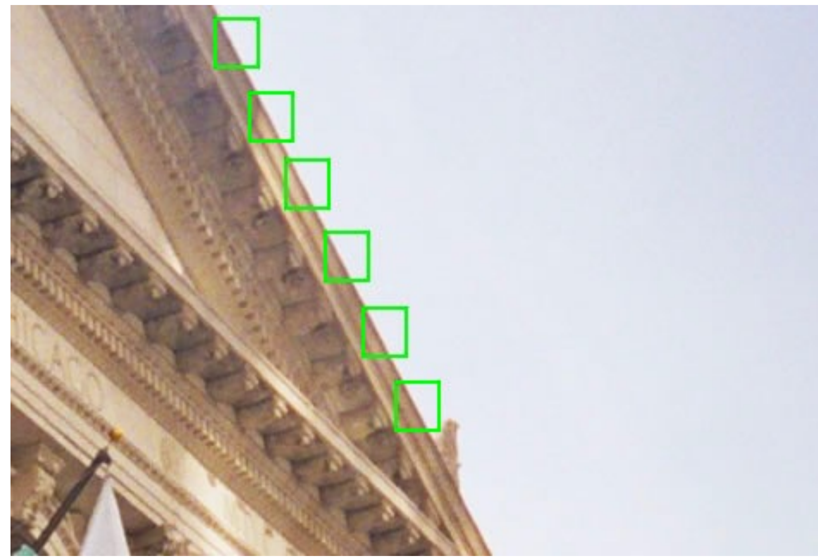
Feature Tracking

- 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.
  - points may appear or disappear
    - need to be able to add/delete tracked points
  - small errors can accumulate as appearance model is updated

Feature Tracking

- What are good features to track?
  - smooth regions and edges?

smooth regions



edges

Bad features! They're not uniquely identifiable!

Feature Tracking

- What are good features to track?
  - Think about what you learnt earlier in this class
  - *Corners!!! (e.g. use Harris or Shi-Tomasi corner detector)*

Feature Tracking

- Recap
  - Harris Detector
    - change of intensity for the shift (u, v):

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Window Function

Shifted Intensity

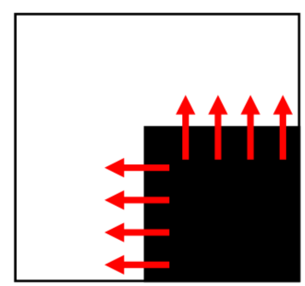
Original Intensity

    - use *Taylor* expansion to approximate:

$$E(u,v) \approx [u \;\; v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

where $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

sum over the region



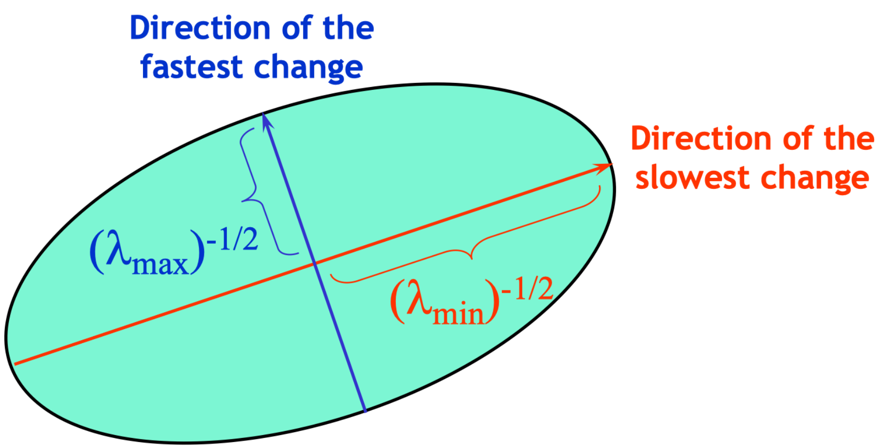
$$M = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

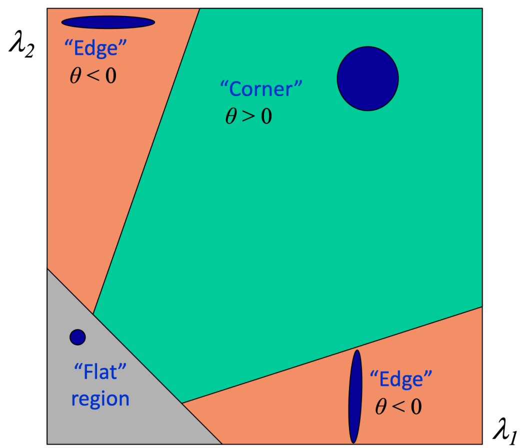
Taylor expansion:

$$I(x+u, y+v) = I(x,y) + uI_x(x,y) + vI_y(x,y) + \cdots$$

Feature Tracking

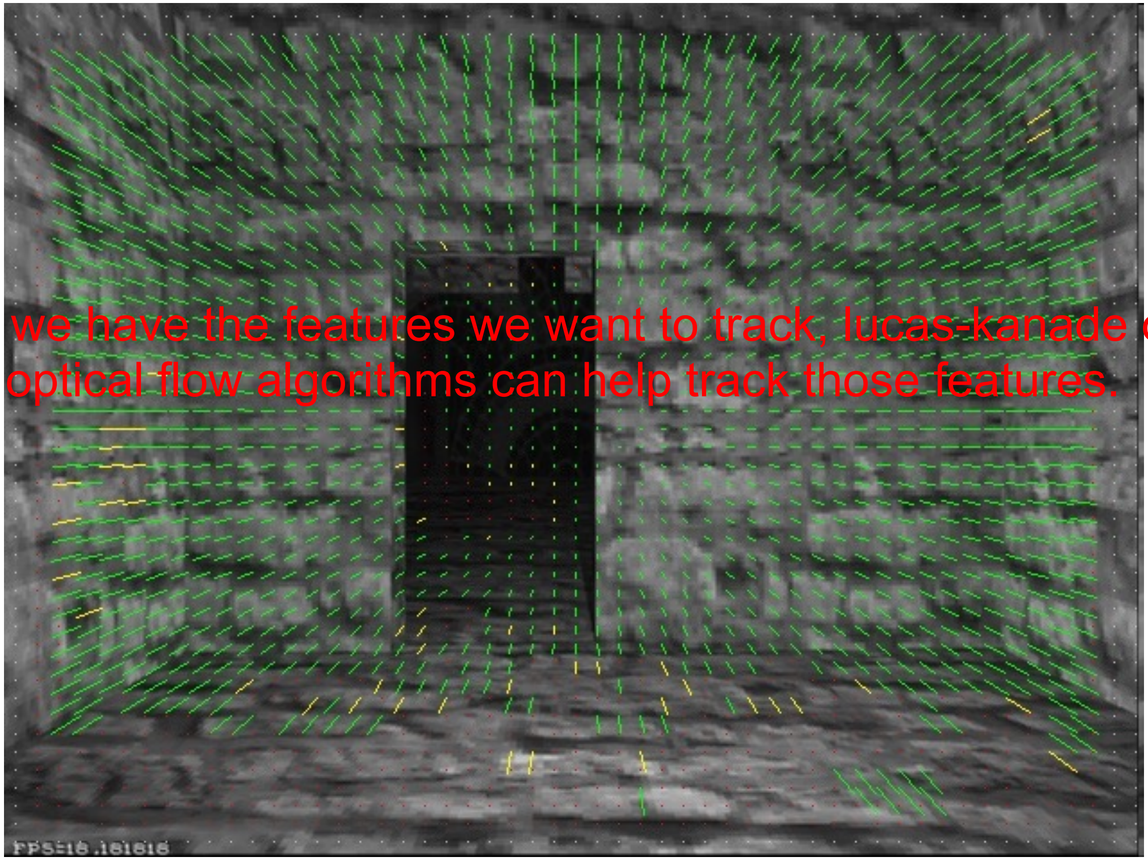
- Recap
  - Harris Detector
    - $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$
      (eigenvalue decomposition)
    - *We can think of M as an ellipse with axis lengths determined by the eigenvalues and orientation determined by R*





Feature Tracking

- How to track the feature points?
  - Optical Flow



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

Picture courtesy of Selim Temizer - Learning and Intelligent Systems (LIS) Group, MIT

What we will learn today?

- Video Tracking
- Feature Tracking
- Simple KLT Tracker
- Higher-Order Motion Model

Simple KLT Tracker

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

Simple KLT Tracker

- KLT tracker for fish

Simple KLT Tracker

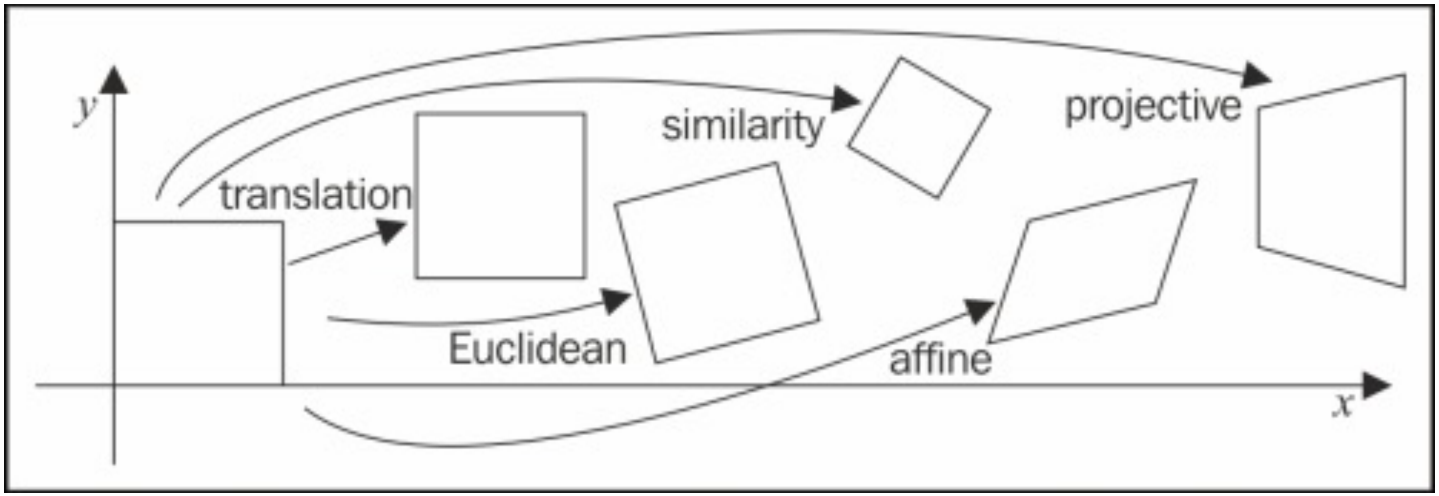
- KLT tracker for beer can

What we will learn today?

- Video Tracking
- Feature Tracking
- Simple KLT Tracker
- Higher-Order Motion Model

Higher-Order Motion Models

- Types of 2D transformations

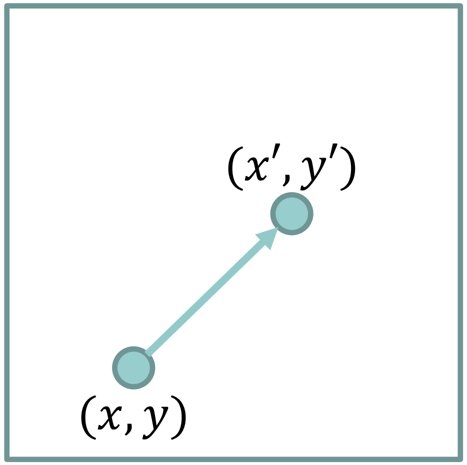


Higher-Order Motion Models

- Translation
  - We can write the transformations as:

$$x' = x + b_1$$

$$y' = y + b_2$$

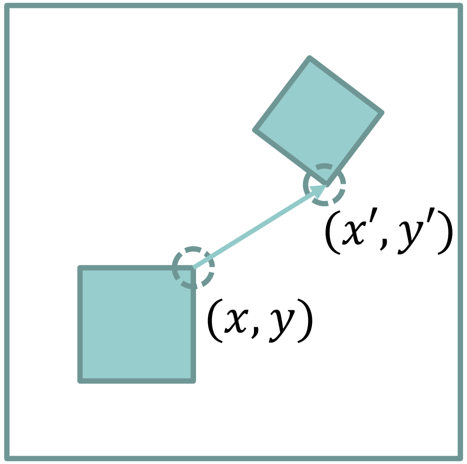


Higher-Order Motion Models

- Similarity
  - Similarity transformation includes uniform *scaling* + *translation* + *rotation*
  - We can write the transformations as:

$$y' = a(x sin\theta + y cos\theta) + b_2$$

$$x' = a(x cos\theta - y sin\theta) + b_1$$

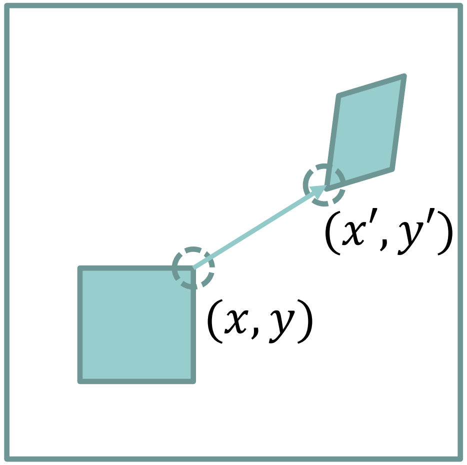


Higher-Order Motion Models

- Affine
  - Affine transformation includes *scaling* + *rotation* + *translation*
  - We can write the transformations as:

$$y' = a_3 x + a_4 y + b_2$$

$$x' = a_1 x + a_2 y + b_1$$



Higher-Order Motion Models

- Recap
  - optical flow model
    - brightness constancy assumption:

      $$I(x, y, t) = I(x + u_1, y + u_2, t + 1)$$

    - use *Taylor* expansion to approximate:

      $$I(x + u, y + v, t + 1) - I(x, y, t) \approx u_1 I_x + u_2 I_y + I_t = 0$$

      which can be written as $0 = \vec{\mathrm{u}}(x, y)^T \vec{\nabla} I + I_t$

    - It measures constant motion (or *translation*)

Higher-Order Motion Models

- What about affine motion?
  - affine flow at location $\vec{x}_0$ is given by:

$$\vec{\mathrm{u}}(\vec{x}) = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} (\vec{x} - \vec{x}_0) + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = A(\vec{x};\vec{x}_0)\vec{p}$$

where $\vec{p} = [a_1\ a_2\ a_3\ a_4\ b_1\ b_2]^{\,T}$ and

$$A(\vec{x};\vec{x}_0) = \begin{bmatrix} x - x_0 & y - y_0 & 0 & 0 & 1 & 0 \\ 0 & 0 & x - x_0 & y - y_0 & 0 & 1 \end{bmatrix}$$

  - substituting into brightness constancy assumption:

$$\vec{\mathrm{u}}(x,y)^T \vec{\nabla} I + I_t = \vec{p}^T A(\vec{x};\vec{x}_0)^T \vec{\nabla} I + I_t = 0$$

Higher-Order Motion Models

- What about similarity motion?
  - try to derive it by yourself

What we have learned today?

- What's video tracking and why do that?
- How to do feature tracking?
  - Detect corners (e.g. Harris corner)
  - Track these corners (e.g. KLT tracker)
- Higher-order motion model
  - similarity motion
  - affine motion
- Further reading
  - http://www.cs.toronto.edu/pub/jepson/teaching/vision/2503/opticalFlow.pdf
  - Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm