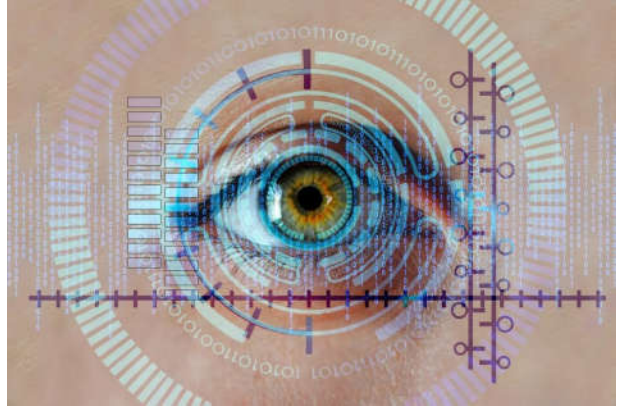
北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Computer Vision



Lecture 16 Tracking

School of Computer Science and Technology

Ying Fu

fuying@bit.edu.cn

Outline

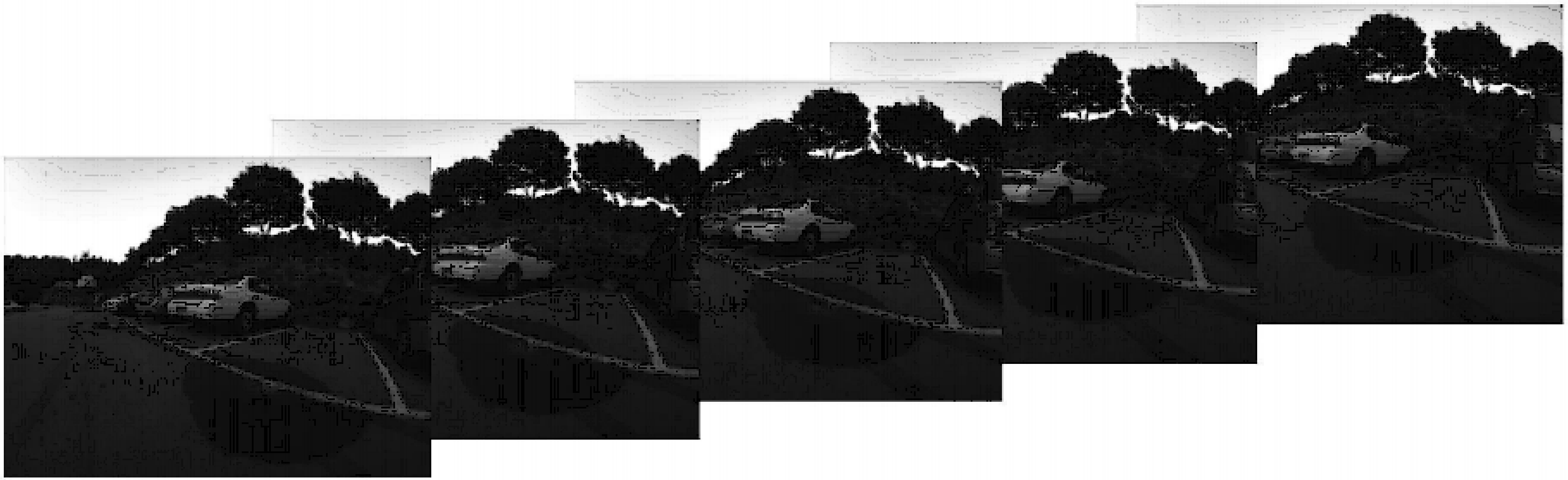
- Feature tracking
- Simple KLT tracker
- 2D transformations (recap)
- Iterative KLT tracker

Outline

- Feature tracking
  - Problem statement
  - Overview

Problem statement

- Image sequence

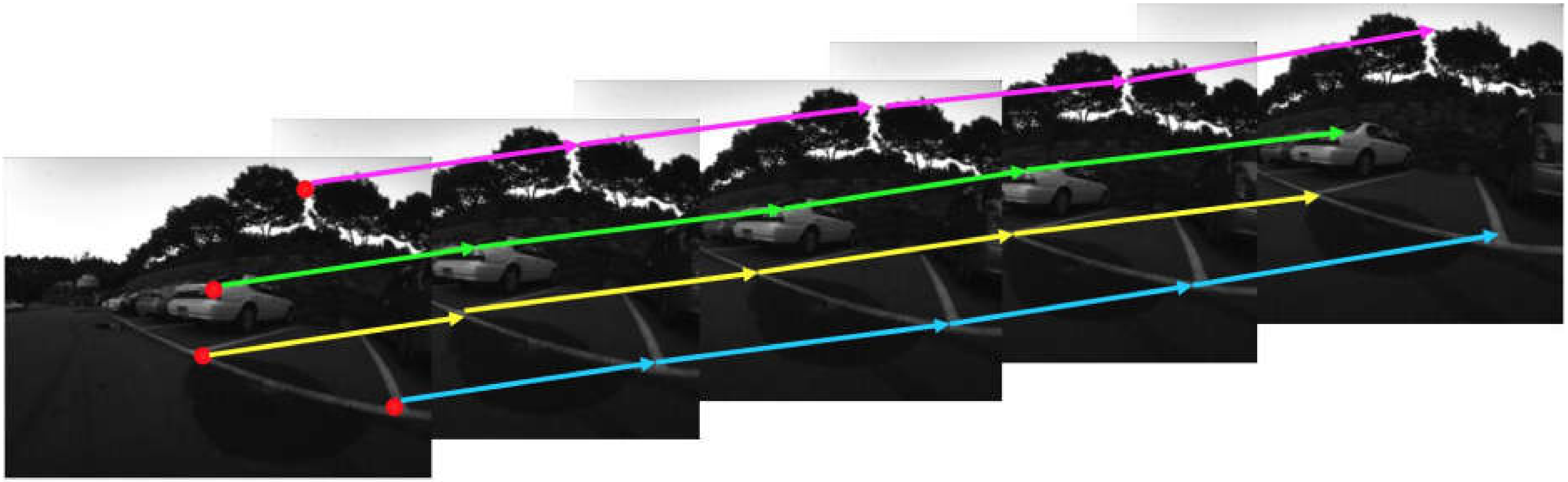


Problem statement

- Feature point detection

Problem statement

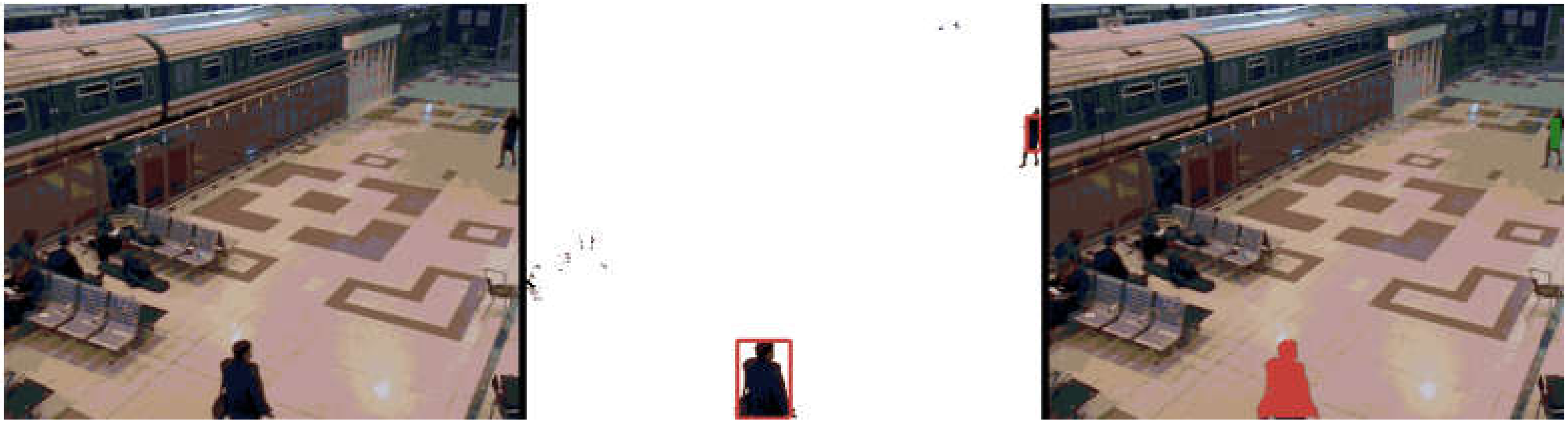
- Feature point tracking



Single object tracking

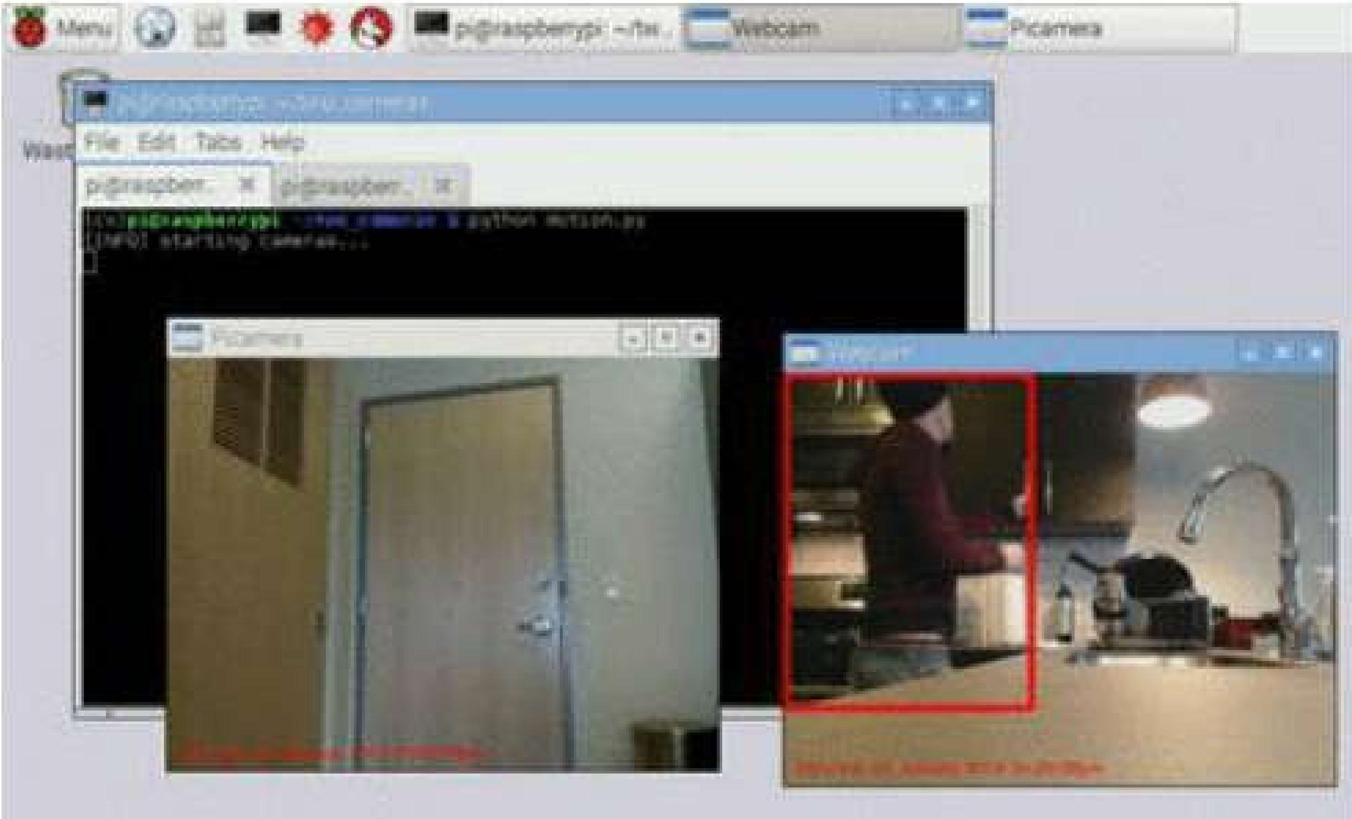
Multiple object tracking

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 is 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!

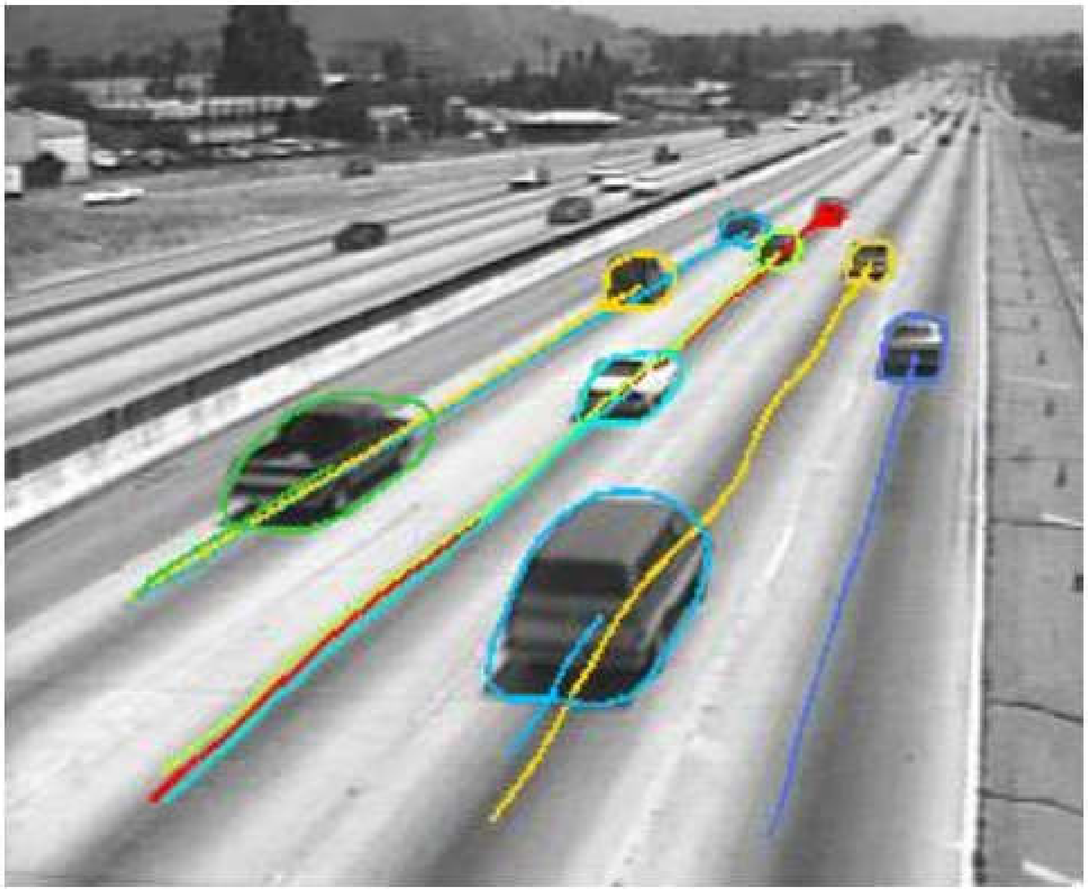
Motion estimation techniques

- Optical flow
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)

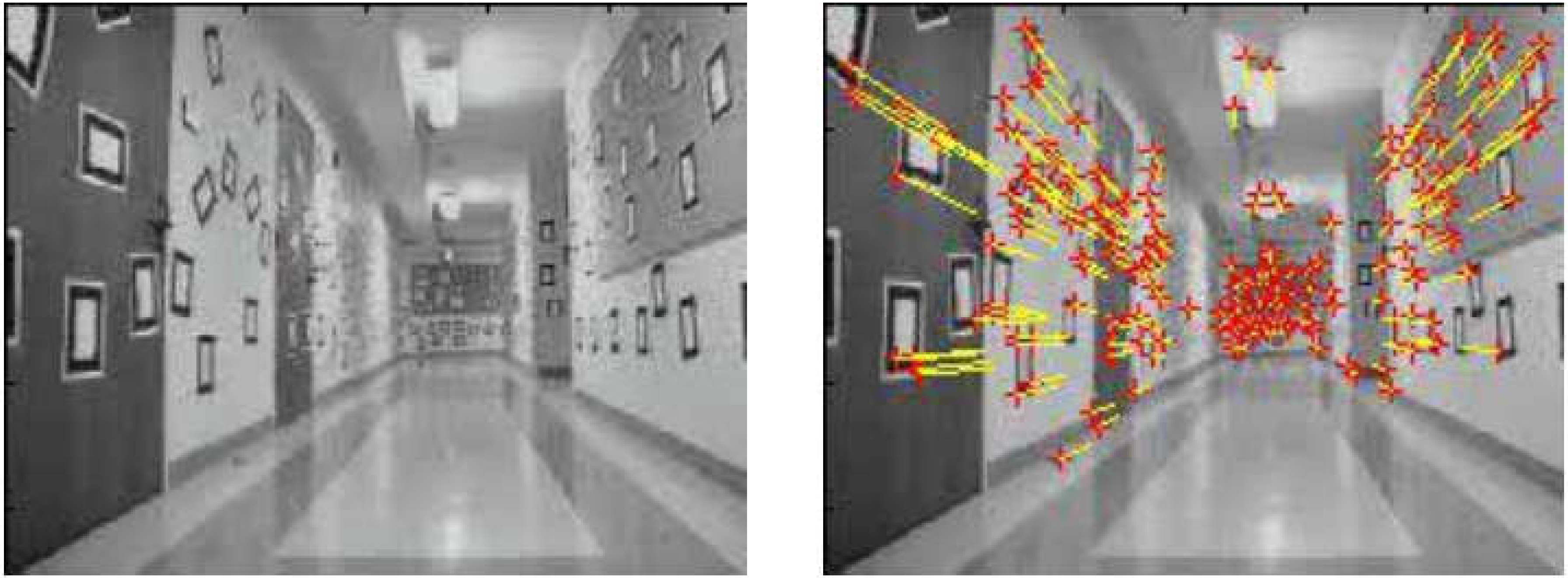
- Feature-tracking
  - Extract visual features (corners, textured areas) and “track” them over multiple frames

Optical flow can help track features

- Once we have the features we want to track, Lucas-Kanade or other optical flow algorithms can help track those features



Feature-tracking



Outline

- Feature tracking
- Simple KLT tracker

  (Kanade–Lucas–Tomasi feature tracker)
  - Pipeline
  - Results

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.

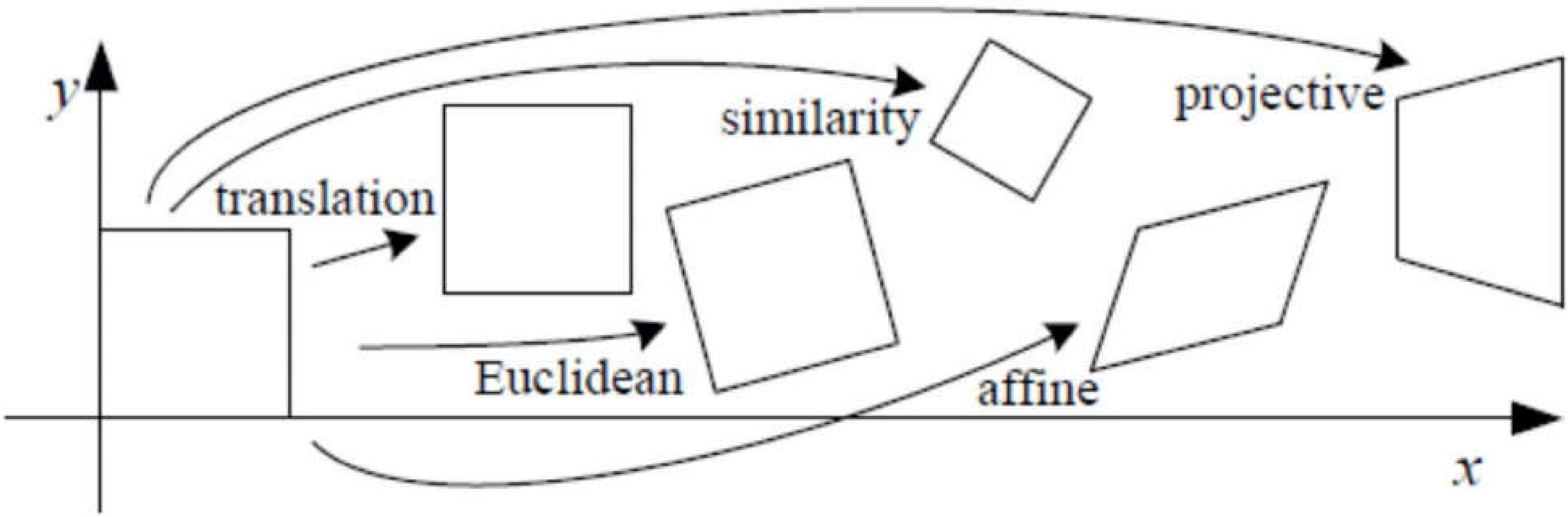
Tracking cars

Tracking movement

Outline

- Feature tracking
- Simple KLT tracker
- 2D transformations (recap)

Types of 2D transformations

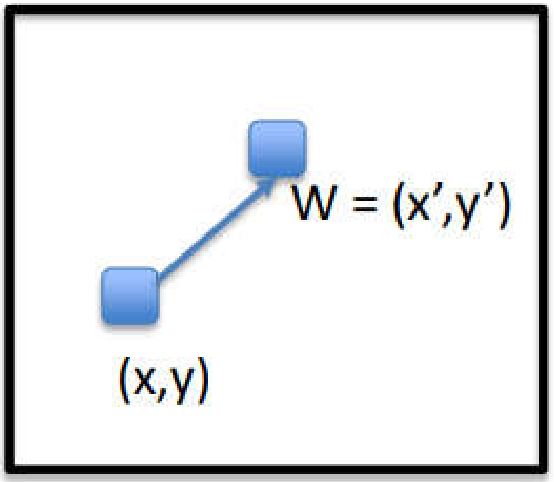


Translation

- 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:

$$x' = x + b_1$$

$$y' = y + b_2$$

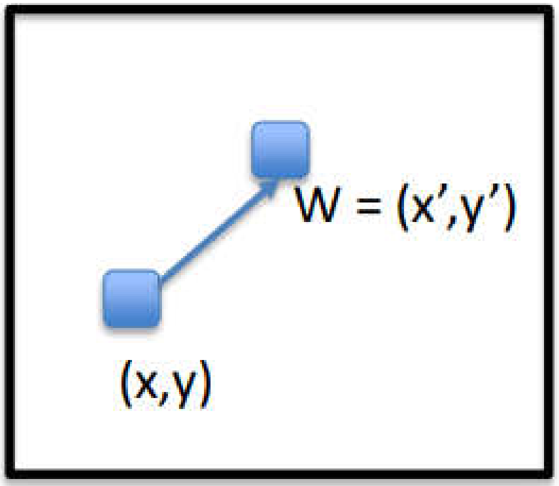


- We can write this as a matrix transformation using homogeneous coordinates:

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

Displacement Model for Translation

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



- There are only two parameters: $\boldsymbol{p}=\begin{bmatrix}b_1\\b_2\end{bmatrix}$

- The derivative of the transformation w.r.t. $\boldsymbol{p}$:

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

- This is called the Jacobian.

Similarity motion

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

$$x' = ax + b_1$$

$$y' = ay + b_2$$

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

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

Affine motion includes scaling + rotation + translation

- Affine motion includes scaling + rotation + translation.

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

$$y' = a_1 x + a_2 y + b_2$$

$$W(\boldsymbol{x};\boldsymbol{p})=\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \boldsymbol{p}=[a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2]^T$$

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

Outline

- Feature tracking
- Simple KLT tracker
- 2D transformations (recap)
- Iterative KLT tracker

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 $\boldsymbol{x}$.
- For each feature at location $\boldsymbol{x}$=[x y]T:
  - Create an initial template for that feature: $T(\boldsymbol{x})$.
  - $T(\boldsymbol{x})$ is usually an image patch around $\boldsymbol{x}$.
- Goal: find new location of feature $\boldsymbol{x}$ at the next frame.
- We will assume $\boldsymbol{x}$ undergoes a transformation (translation, affine, …) parametrized by $\boldsymbol{p}$ to reach its new location $W(\boldsymbol{x};\boldsymbol{p})$.

KLT objective

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

$$\sum_{x}\left[I\big(W(\boldsymbol{x};\boldsymbol{p})\big) - T(\boldsymbol{x})\right]^2$$

- $W(\boldsymbol{x};\boldsymbol{p})$ is the new location of feature $\boldsymbol{x}$.
- $I(W(\boldsymbol{x};\boldsymbol{p}))$ is image intensity at the new location.
- Recall that $\boldsymbol{p}$ is our vector of parameters that define the transformation that took $\boldsymbol{x}$ to its new location $W(\boldsymbol{x};\boldsymbol{p})$.
- Sum is over an image patch around $\boldsymbol{x}$.

KLT objective

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

$$\sum_{x}\left[I\big(W(\boldsymbol{x};\boldsymbol{p})\big)-T(\boldsymbol{x})\right]^2$$

- We will instead break down $\boldsymbol{p}=\boldsymbol{p_0}+\Delta\boldsymbol{p}$
  - Large + small/residual motion
  - Where $\boldsymbol{p_0}$ is going to be fixed and we will solve for $\Delta\boldsymbol{p}$, which is a small value.
  - We can initialize $\boldsymbol{p_0}$ with our best guess of what the motion is, and then calculate $\Delta\boldsymbol{p}$.
- We can substitute $\boldsymbol{p}$ to get:

$$\sum_{x}\left[I\big(W(\boldsymbol{x};\boldsymbol{p_0}+\Delta\boldsymbol{p})\big)-T(\boldsymbol{x})\right]^2$$

Taylor series

- Taylor series is defined as

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

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

Expanded KLT objective

$$\sum_{x}\left[I\big(W(\boldsymbol{x};\boldsymbol{p_0}+\Delta\boldsymbol{p})\big)-T(\boldsymbol{x})\right]^2$$

$$\approx\sum_{x}\left[I\big(W(\boldsymbol{x};\boldsymbol{p_0})\big)+\nabla I\frac{\partial W}{\partial \boldsymbol{p}}\Delta\boldsymbol{p}-T(\boldsymbol{x})\right]^2$$

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

Expanded KLT objective

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

$$\underset{\Delta\boldsymbol{p}}{\operatorname{argmin}} \sum_{x} \left[I\big(W(\boldsymbol{x};\boldsymbol{p_0})\big) + \nabla I \frac{\partial W}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right]^2$$

- Where $\nabla I = \begin{bmatrix} I_x & I_y \end{bmatrix}$
- Differentiate wrt $\Delta\boldsymbol{p}$ and setting it to zero

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

Solving for $\Delta \boldsymbol{p}$

- Solving for $\Delta \boldsymbol{p}$ in

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

- We get

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

- where

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

- $\boldsymbol{H}$ must be invertible!

Interpreting the H matrix for translation transformations

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

Recall that

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

Therefore,

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

$$= \sum_{\boldsymbol{x}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

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

Interpreting the H matrix for affine transformations

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

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

Given the features from Harris detector:

- 1.Initialize $\boldsymbol{p_0}$.
- 2.Compute the initial templates $T(\boldsymbol{x})$ for each feature.
- 3.Transform the features in the image I with $W(\boldsymbol{x}; \boldsymbol{p_0})$.
- 4.Measure the error: $I\big(W(\boldsymbol{x}; \boldsymbol{p_0})\big) - T(\boldsymbol{x})$.
- 5.Compute the image gradients $\nabla I = \left[I_x \;\; I_y \right]$.
- 6.Evaluate the Jacobian $\frac{\partial W}{\partial \boldsymbol{p}}$.
- 7.Compute steepest descent $\nabla I \frac{\partial W}{\partial \boldsymbol{p}}$.
- 8.Compute Inverse Hessian H^{-1}
- 9.Calculate the change in parameters $\Delta \boldsymbol{p}$
- 10.Update parameters $\boldsymbol{p_0}$= $\boldsymbol{p_0}$+$\boldsymbol{p}$
- 11.Repeat 3 to 10 until $\Delta \boldsymbol{p}$ 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 (size of neighborhood/template around x)
  - 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)

References

- Basic reading:
  - Szeliski textbook, Section 9.3, 9.4