Lecture 18: Tracking

Khaled Jedoui, Jamie Xie, Michelle Guo, Nikhil Cheerla.
Department of Computer Science
Stanford University
Stanford, CA 94305
{thekej, jaimiex, ncheerla, mguo95}@stanford.edu

1 Introduction: What is tracking?

1.1 Definition

Visual tracking is the process of locating a moving object (or multiple objects) over time in a sequence.

1.2 Objective

The objective of tracking is to associate target objects and estimate target state over time in consecutive video frames.

1.3 Applications

Tracking has a variety of application, some of which are:

- Human-computer interaction: Eye Tracking Systems[1]
- Security and surveillance[3]
- Augmented reality[2]
- Traffic control: Road transportation systems[5]
- Medical imaging[4]

1.4 Challenges

Note, that tracking can be a time consuming process due to the amount of data that in video. Besides, tracking relies on object recognition algorithms for tracking, which might become more challenging and prone to failure for the following reasons:

- Variations due to geometric changes like the scale of the tracked object
- Changes due to illumination and other photometric aspects
- Occlusions in the image frame
- Motion that is non-linear
- Blurry videos or videos with bad resolution might make the recognition fail
- Similar objects in the scene

2 Feature Tracking

2.1 Definition

Feature tracking is the detection of visual feature points (corners, textured areas, ...) and tracking them over a sequence of frames (images).

2.2 Challenges of 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

2.3 What are good features to track?

What kinds of image regions can we detect easily and consistently? We need a way that can measure "quality" of features from just a single image. Intuitively, we want to avoid smooth regions and edges. A solution for such a problem is to use Harris Corners. Detecting Harris corners as our key points to track guarantees small error sensitivity!

Once we detect the features we want to track, we can use optical flow algorithms to solve our motion estimation problem and track our features.

2.4 Example

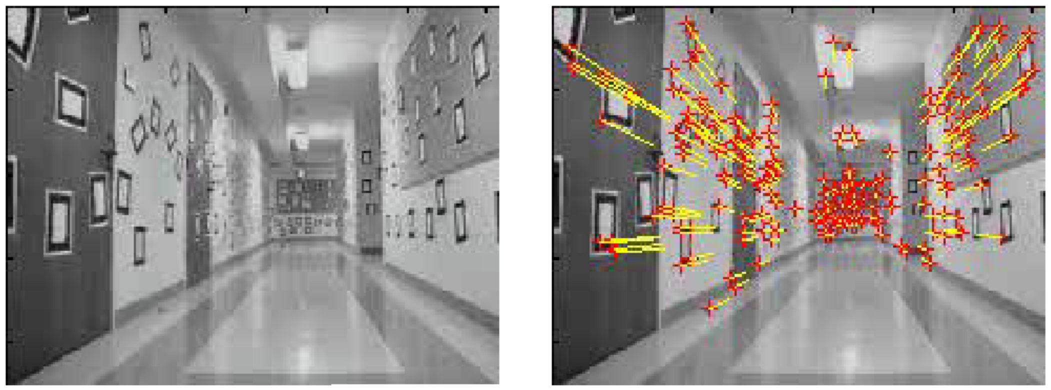

Figure 1: Courtesy of Jean-Yves Bouguet–Vision Lab, California Institute of Technology

2.5 Tracking methods

2.5.1 Simple Kanade–Lucas–Tomasi feature tracker

The Kanade–Lucas–Tomasi (KLT) feature tracker is an approach to feature extraction. KLT makes use of spatial intensity information to direct the search for the position that yields the best match. Its algorithm is:

1. Find a good point to track (harris corner)
   - Harris corner points have sufficiently large eigenvalues, so the optical flow equation is solvable.
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

    - If the patch around the new point differs sufficiently from the old point, we discard these points.

4. Introduce new Harris points by applying Harris detector at every (10 or 15) frames
5. Track new and old Harris points using steps 2-3

In the following frames from tracking videos, arrows represent the tracking motion of the harris corners.

Figure 2: Frames from fish-tracking video, courtesy of Kanade

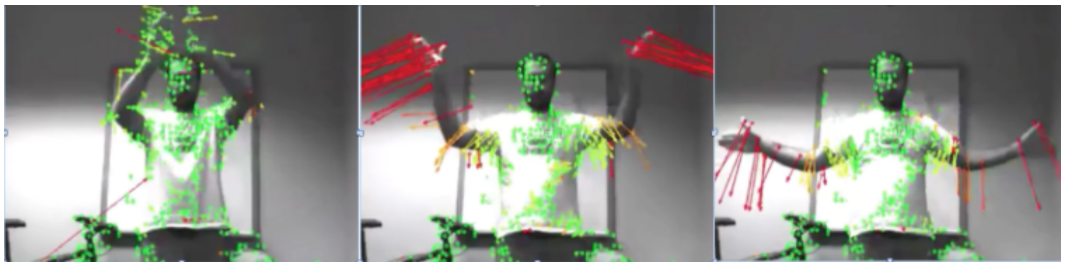
Figure 3: Frames from man-tracking video, courtesy of Kanade

3 2D Transformations

3.1 Types of 2D Transformations

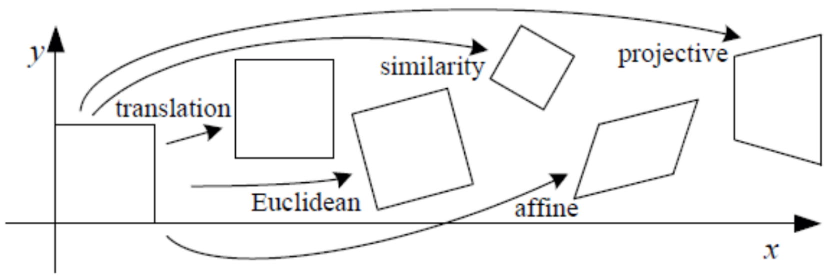

Figure 4: Types of 2D transformations.

There are several types of 2D transformations. Choosing the correct 2D transformations can depend on the camera (e.g. placement, movement, and viewpoint) and objects. A number of 2D transformations are shown in Figure 3.1. Examples of 2D transformations include:

- **Translation Transformation**. (e.g. Fixed overhead cameras)
- **Similarity Transformation**. (e.g. Fixed cameras of a basketball game)
- **Affine Transformation**. (e.g. People in pedestrian detection)
- **Projective Transformation**. (e.g. Moving cameras)

In this section we will cover three common transformations: translation, similarity, and affine.

3.2 Translation

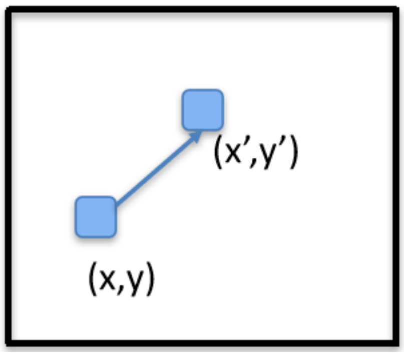

Figure 5: Translation

Translational motion is the motion by which a body shifts from one point in space to another. Assume we have a simple point m with coordinates (x, y). Applying a translation motion on m shifts it from (x, y) to (x', y') where

$$\begin{aligned} x' &= x + b_1 \\ y' &= y + b_2 \end{aligned} \tag{1}$$

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

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2}$$

Let W be the above transformation defined as:

$$W(\boldsymbol{x}; \boldsymbol{p}) = \begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \end{pmatrix} \tag{3}$$

where the parameter vector is $p = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$.

Taking the partial derivative of W with respect to $\boldsymbol{p}$ we get:

$$\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x}; \boldsymbol{p}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

This is called the Jacobian.

3.3 Similarity Motion

Similarity motion is a rigid motion that includes scaling and translation.

We can define the similarity as:

$$\begin{aligned} x' &= ax + b_1 \\ y' &= ay + b_2 \end{aligned} \tag{5}$$

The similarity transformation matrix W and parameters p are defined as the following:

$$\begin{aligned} W &= \begin{pmatrix} a & 0 & b_1 \\ 0 & a & b_2 \end{pmatrix} \\ \boldsymbol{p} &= \begin{pmatrix} a & b_1 & b_2 \end{pmatrix}^T \end{aligned} \tag{6}$$

The Jacobian of the similarity transformation is then:

$$\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x}; \boldsymbol{p}) = \begin{pmatrix} x & 1 & 0 \\ y & 0 & 1 \end{pmatrix} \tag{7}$$

3.4 Affine motion

Affine motion includes scaling, rotation, and translation. We can express this as the following:

$$\begin{aligned} x' &= a_1 x + a_2 y + b_1 \\ y' &= a_3 x + a_4 y + b_2 \end{aligned} \tag{8}$$

The affine transformation can be described with the following transformation matrix W and parameters p:

$$\begin{aligned} W &= \begin{pmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \end{pmatrix} \\ p &= (a_1 \quad a_2 \quad b_1 \quad a_3 \quad a_4 \quad b_2)^T \end{aligned} \tag{9}$$

Finally, the Jacobian for affine motion is the following:

$$\frac{\partial W}{\partial \boldsymbol{p}}(\boldsymbol{x};\boldsymbol{p}) = \begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{pmatrix} \tag{10}$$

4 Iterative KLT tracker

4.1 Problem formulation

Given a video sequence, find the sequence of transforms that maps each frame to the next frame. Should be able to deal with arbitrary types of motion, including object motion and camera/perspective motion.

4.2 Approach

This approach differs from the simple KLT tracker by the way it links frames: instead of using optical-flow to link motion vectors and track motion, we directly solve for the relevant transforms using feature data and linear approximations. This allows us to deal with more complex (such as affine and projective) transforms and link objects more robustly.

Steps:

1. First, use Harris corner detection to find the features to be tracked.
2. For each feature at location $x = [x, y]^T$: Choose a feature descriptor and use it to create an initial template for that feature (likely using nearby pixels): $T(x)$.
3. Solve for the transform p that minimizes the error of the feature description around $x_2 = W(x; p)$ (your hypothesis for where the feature's new location is) in the next frame. In other words, solve the equation
$$\sum_x [T(W(x;p)) - T(x)]^2$$
4. Iteratively reapply this to link frames together, storing the coordinates of the features as the transforms are continuously applied. This should give you a measure of how objects move through frames.
5. Just as before, every 10-15 frames introduce new Harris corners to account for occlusion and "lost" features.

4.3 Math

We can in fact analytically derive an approximation method for finding p (in Step 3). Assume that you have an initial guess for p, p_0, and $p = p_0 + \Delta p$.

Now,

$$E = \sum_x [T(W(x;p)) - T(x)]^2 = \sum_x [T(W(x;p_0 + \Delta p)) - T(x)]^2$$

But using the Taylor approximation, we see that this error term is roughly equal to :

$$E \approx \sum_x [T(W(x;p_0)) + \nabla T \frac{\partial W}{\partial p} \Delta p - T(x)]^2$$

To minimize this term, we take the derivative with regard to p_0 and set it equal to 0, then solve for p_0.

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

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

, where H is $\sum_x [\nabla T \frac{\partial W}{\partial p}]^T [\nabla T \frac{\partial W}{\partial p}]$

By iteratively setting $p_0 = p_0 + \Delta p$, we can eventually converge on an accurate, error-minimizing value of p, which tells us what the transform is.

4.4 Link to Harris Corner Detection

For translation motion,

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

, so it is easy to show that

$$H = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}$$

. However, the Harris corner detector assumes that whenever H has large eigenvalues (i.e it is stably invertible), then the point within the window is a corner. Thus, corners tend to be good features for computing translations, precisely because the resulting matrices produced by corners are stably invertible.

References

[1] S. Chandra, G. Sharma, S. Malhotra, D. Jha, and A. P. Mittal. Eye tracking based human computer interaction: Applications and their uses. pages 1–5, Dec 2015.

[2] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, July 2006.

[3] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, and S. Pankanti. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, March 2005.

[4] P. Mountney, D. Stoyanov, and G. Z. Yang. Three-dimensional tissue deformation recovery and tracking. *IEEE Signal Processing Magazine*, 27(4):14–24, July 2010.

[5] O. Rostamianfar, F. Janabi-Sharifi, and I. Hassanzadeh. Visual tracking system for dense traffic intersections. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 2000–2004, May 2006.