CS-E4850 Computer Vision, Answers to Exercise Round 8

Adam Ilyas 725819

December 14, 2018

Exercise 1. Face tracking example using KLT tracker.
This exercise is based on the python face tracking demo `faceTrackingDemo.py`
Run the example as instructed below and answer the questions.

a) Run the python example script with the command `python faceTrackingDemo.py`

b) Run the python example script with different input `python faceTrackingDemo.py --input=./obama.avi`

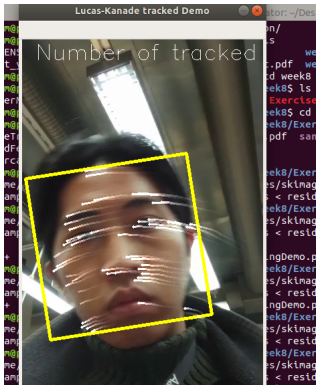
c) What could be the main reasons why most of the features are not tracked very long in case b) above?

Ans We notice that when the image is rotated, the number of tracked features dropped by half. Further more, when the speed of movement of the camera increase, less features get tracked because the points move to far at a high speed.

d) How one could try to avoid the problem that the features are gradually lost? Suggest a one or more improvements.

Ans. Try to restrict rotation and curved movements. We try to reduce large movements over a short period of time

e) Voluntary task: Capture a video of your own face or of a picture of a face, and check that whether the tracking works for you. That is, replace `obama.avi` with your own video.



Exercise 2. Kanade-Lucas-Tomasi (KLT) feature tracking
Read Sections 2.1 and 2.2 on pages 2 and 3 from the paper `BakerMatthews.pdf`, which is given in `Exercise08.zip`. Show that the Equation (10) in the paper gives the same solution as the equations on slide 25 of Lecture 7, when the geometric warping W (between the current frame and the template window in the previous frame) is a translation.

A few things we should know:
Brightness Constancy Equation:

$$\begin{aligned} I(x, y, t-1) &= I(x + u(x,y), y + v(x,y), t) \\ &\approx I(x, y, t) + I_x u(x,y) + I_y v(x,y) \end{aligned}$$

Hence,

$$I_x u + I_y v + I_t \approx 0$$

The brightness constancy constraint

$$\nabla I \cdot (u, v) + I_t = 0$$

To get more equations for a pixel for the aperture problem (Spatial coherence constraint: pretend the pixels neighbors have the same (u,v))

$$\nabla I(x_i) \cdot [u, v] + I_t(x_i) = 0$$

$$\begin{bmatrix} I_x(x_1) & I_x(x_1) \\ I_x(x_2) & I_x(x_2) \\ \vdots & \vdots \\ I_x(x_n) & I_x(x_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} I_t(x_1) \\ I_t(x_2) \\ \vdots \\ I_t(x_n) \end{bmatrix}$$

$$\underset{n\times 2}{A}\ \underset{2\times 1}{d} = \underset{n\times 1}{b}$$

We can find the solution by Lucas-Kanade flow

$$(A^\intercal A)d = A^\intercal b$$
$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Ans. Based on paper `BakerMatthews.pdf`, We let $W(x;p)$ denote the parameterized set of allowed warps, where $p = (p_1, \ldots, p_n)^\intercal$ is a vector of parameters.

The warp $W(x;p)$ takes the pixel x in the coordinate frame of the template T and maps it to the sub-pixel location $W(x;p)$ in the coordinate frame of the image I .

For the case of optical flow, our warp is the following translation:

$$W(x;p) = \begin{bmatrix} x+u \\ y+v \end{bmatrix}, \ \Delta p = \begin{bmatrix} u \\ v \end{bmatrix},$$

The Jacobian of the warp is

$$\frac{\delta W}{\delta p} = \begin{bmatrix} \delta W_x/\delta u & \delta W_x/\delta v \\ \delta W_y/\delta u & \delta W_y/\delta v \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$\frac{\delta W}{\delta p}$ actually an identity matrix. We can use this in equation 10

$$\Delta p = H^{-1} \sum_x \Big[\nabla I \frac{\delta W}{\delta p} \Big]^\intercal [T(x) - I(W(x;p))]$$

where

$$\begin{aligned} H &= \sum_x \big[\nabla I \frac{\delta W}{\delta p} \big]^\intercal \big[\nabla I \frac{\delta W}{\delta p} \big] \\ &= \sum_x \frac{\delta W}{\delta p}^\intercal \nabla I^\intercal \nabla I \frac{\delta W}{\delta p} \\ &= \sum_x \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\delta I}{\delta x} \\ \frac{\delta I}{\delta y} \end{bmatrix} \begin{bmatrix} \frac{\delta I}{\delta x} & \frac{\delta I}{\delta y} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \sum \begin{bmatrix} \frac{\delta I}{\delta x}\frac{\delta I}{\delta x} & \frac{\delta I}{\delta x}\frac{\delta I}{\delta y} \\ \frac{\delta I}{\delta y}\frac{\delta I}{\delta x} & \frac{\delta I}{\delta y}\frac{\delta I}{\delta y} \end{bmatrix} \\ &= \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \end{aligned}$$

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

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

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \Delta p = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [T(x) - I(W(x;p))]$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [T(x) - I(W(x;p))]$$

Note that the template $T(x)$ is an extracted sub-region of the image at $t = 1$ and $I(x)$ is the image at $t = 2$. Hence,

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [-I_t]$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Which is the summation over all pixels in the window.