Tracking (Lucas Kanade)

CAP 5415, FALL 2019
References:

• Lecture notes of Robert Collins (Penn State University).

Edges and Features are precursors to other functions such as tracking, recovering 3D shape from 2D images, and object classification.
Examples Entity Tracking
Example – Point/Feature Tracking
The sum of squared differences (SSD) algorithm can find where the image patch has moved in the scene
$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2$

current frame

$u,v = \text{hypothesized location of template in current frame}$

template (model)
Basic KLT for templates

\[ E(u, v) = \sum [I(x + u, y + v) - T(x, y)]^2 \]

\[ \approx \sum [I(x, y) + uI_x(x, y) + vI_y(x, y) - T(x, y)]^2 \quad \text{First order approx} \]

\[ = \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]^2 \]

Take partial derivs and set to zero

\[ \frac{\delta E}{du} = \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]I_x(x, y) = 0 \]

\[ \frac{\delta E}{dv} = \sum [uI_x(x, y) + vI_y(x, y) + D(x, y)]I_y(x, y) = 0 \]

Form matrix equation

\[ \sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \sum \begin{bmatrix} I_xD \\ I_yD \end{bmatrix} \quad \text{solve via least-squares} \]
Limitation of simple template matching

Assumption of constant flow (pure translation) for all pixels in a larger window is unreasonable for long periods of time.

However, we can easily generalize Lucas-Kanade approach to other 2D parametric motion models (like affine or projective) by introducing a “warp” function $W$.

$$E(u,v) = \sum [I(x+u,y+v) - T(x,y)]^2 \quad \text{generalize} \quad \sum [I(W([x,y]; P)) - T([x,y])]^2$$
Definition of 2D image warp functions

Translation. 2D translations can be written as \( x' = x + t \) or

\[
x' = \begin{bmatrix} 1 & t \\ \end{bmatrix} x
\]

(2.14)

where \( I \) is the \((2 \times 2)\) identity matrix or

\[
x' = \begin{bmatrix} I & t \\ 0^T & 1 \\ \end{bmatrix} x
\]

(2.15)

where 0 is the zero vector. Using a \( 2 \times 3 \) matrix results in a more compact notation, whereas using a full-rank \( 3 \times 3 \) matrix (which can be obtained from the \( 2 \times 3 \) matrix by appending a \([0^T 1]\) row) makes it possible to chain transformations using matrix multiplication. Note that in any equation where an augmented vector such as \( x \) appears on both sides, it can always be replaced with a full homogeneous vector \( \bar{x} \).

Rotation + translation. This transformation is also known as 2D rigid body motion or the 2D Euclidean transformation (since Euclidean distances are preserved). It can be written as \( x' = Rx + t \) or

\[
x' = \begin{bmatrix} R & t \\ \end{bmatrix} x
\]

(2.16)

where

\[
R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \\ \end{bmatrix}
\]

(2.17)

is an orthonormal rotation matrix with \( R R^T = I \) and \( |R| = 1 \).

Scaled rotation. Also known as the similarity transform, this transformation can be expressed as \( x' = sRx + t \) where \( s \) is an arbitrary scale factor. It can also be written as

\[
x' = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ \end{bmatrix} x
\]

(2.18)

where we no longer require that \( a^2 + b^2 = 1 \). The similarity transform preserves angles between lines.

Affine. The affine transformation is written as \( x' = Ax \), where \( A \) is an arbitrary \( 2 \times 3 \) matrix, i.e.,

\[
x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ \end{bmatrix} x.
\]

(2.19)

Parallel lines remain parallel under affine transformations.

Projective. This transformation, also known as a perspective transform or homography, operates on homogeneous coordinates,

\[
x' = \bar{H}\bar{x}
\]

(2.20)

where \( \bar{H} \) is an arbitrary \( 3 \times 3 \) matrix. Note that \( \bar{H} \) is homogeneous, i.e., it is only defined up to a scale, and that two \( \bar{H} \) matrices that differ only by scale are equivalent. The resulting homogeneous coordinate \( \bar{x}' \) must be normalized in order to obtain an inhomogeneous result \( x' \), i.e.,

\[
x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad \text{and} \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}.
\]

(2.21)

Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).

Warp functions

- $P$ is a vector of parameters that define the image warp
- $P$ has just 2 elements for pure translation.
- Rotation has 3 degrees of freedom (DoF), similarity has 4
- Affine warp has 6 parameters $p = [p_1 \ p_2 \ p_6]$

Translation warp

Affine warp

$$W(x; p) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$ \hspace{2cm} (1)

$$W(x; p) = \begin{pmatrix} (1 + p_1) \cdot x + p_3 \cdot y + p_5 \\ p_2 \cdot x + (1 + p_4) \cdot y + p_6 \end{pmatrix} = \begin{pmatrix} \frac{1 + p_1}{p_2} \ & \frac{p_3}{p_2} \ & \frac{p_5}{p_2} \\ \frac{1}{1 + p_4} \ & \frac{p_6}{1 + p_4} \ & \frac{1}{1 + p_4} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$ \hspace{2cm} (2)
KLT algorithm summary

• We wish to find the optimum value of $p$ that minimizes $\sum_{x} [I(W(x; p)) - T(x)]^2$.

• Assuming that a current value of $p$ is known, find $\Delta p$ that minimizes
  $$\sum_{x} [I(W(x; p + \Delta p)) - T(x)]^2$$

• Update the current estimate for the parameters $p \leftarrow p + \Delta p$.

• Stop when $\|\Delta p\| \leq \epsilon$. 
Parameter update algorithm-1

\[
\sum_x [I(W(x; p + \Delta p)) - T(x)]^2 = \sum_x \left[ I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2.
\]

\[
\frac{d}{dp} \{I(W(x;p))\} = \nabla I. \frac{\partial W(x;p)}{\partial p} \quad \text{- Application of Chain Rule!}
\]

Also, \[
\frac{d}{dp} \{I(W(x;p))\} = \frac{I(W(x;p+\Delta p))-I(W(x;p))}{\Delta p}
\]

- By definition

Therefore,

\[
\nabla I. \frac{\partial W(x;p)}{\partial p} = \frac{I(W(x;p+\Delta p))-I(W(x;p))}{\Delta p}, \quad \text{or}
\]

\[
I(W(x; p + \Delta p)) = \nabla I. \frac{\partial W(x;p)}{\partial p} \Delta p + I(W(x; p))
\]

Note
Parameter update algorithm 2

The derivative of

\[ \sum_x \left[ I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right]^2 \]

w.r.t. to \( \Delta p \) is

\[ 2 \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ I(W(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x) \right] \]

Setting this to zero we get

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

where

\[ H = \sum_x \left[ \nabla I \frac{\partial W}{\partial p} \right]^T \left[ \nabla I \frac{\partial W}{\partial p} \right]. \]
Parameter update algorithm -3

- What is $\frac{\partial W}{\partial p}$?

- For affine warp

  \[ W(x; p) = \begin{pmatrix} (1 + p_1) \cdot x + p_3 \cdot y + p_5 \\ p_2 \cdot x + (1 + p_4) \cdot y + p_6 \end{pmatrix} \]

  \[ = \begin{pmatrix} 1 + p_1 & p_3 & p_5 \\ p_2 & 1 + p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \]  

- Therefore, by taking derivative w.r.t. to each element of $p$ we get

  \[ \frac{\partial W}{\partial p} = \begin{pmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{pmatrix}. \]
Warp $I$ to obtain $I(W([x,y]; P))$

Compute the error image $T(x) - I(W([x,y]; P))$

Warp the gradient $\nabla I$ with $W([x,y]; P)$

Evaluate $\frac{\partial W}{\partial P}$ at $([x,y]; P)$ (Jacobian)

Compute steepest descent images $\nabla I \frac{\partial W}{\partial P}$

Compute Hessian matrix $\sum (\nabla I \frac{\partial W}{\partial P})^T (\nabla I \frac{\partial W}{\partial P})$

Compute $\sum (\nabla I \frac{\partial W}{\partial P})^T (T(x,y) - I(W([x,y]; P)))$

Compute $\Delta P$

Update $P \leftarrow P + \Delta P$
References:

• Lecture notes of Robert Collins (Penn State University).