Patch tracking based on comparing its pixels

Tomáš Svoboda, svoboda@cmp.felk.cvut.cz
Czech Technical University in Prague, Center for Machine Perception
http://cmp.felk.cvut.cz
Last update: March 23, 2015

Talk Outline

- comparing patch pixels
- normalized cross-correlation, ssd . . .
- KLT - gradient based optimization
- good features to track

1Please note that the lecture will be accompanied be several sketches and derivations on the blackboard and few live-interactive demos in Matlab
What is the problem?
Tracking of dense sequences — camera motion

$T$ - Template

$I$ - Image

Scene static, camera moves.
Tracking of dense sequences — object motion

\(T\) - Template
\(I\) - Image

Camera static, object moves.
Goal is to align a template image $T(x)$ to an input image $I(x)$. $x$ column vector containing image coordinates $[x, y]^\top$. The $I(x)$ could be also a small subwindow within an image.
How to measure the alignment?

- What is the best criterial function?
- How to find the best match, in other words, how to find extremum of the criterial function?
How to measure the alignment?

- What is the best criterial function?

- How to find the best match, in other words, how to find extremum of the criterial function?

Criterial function

What are the desired properties (on a certain domain)?
How to measure the alignment?

- What is the best criterial function?
- How to find the best match, in other words, how to find extremum of the criterial function?

Criterial function

What are the desired properties (on a certain domain)?

- convex (remember the optimization course?)
- discriminative
- ...
Normalized cross-correlation

You may know it as correlation coefficient (from statistics)

\[ \rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \]

where \( \sigma \) means standard deviation.
Normalized cross-correlation

You may know it as correlation coefficient (from statistics)

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

where $\sigma$ means standard deviation.

Having template $T(k,l)$ and image $I(x,y)$,

$$r(x,y) = \frac{\sum_k \sum_l (T(k,l) - \overline{T})(I(x+k,y+l) - \overline{I(x,y)})}{\sqrt{\sum_k \sum_l (T(k,l) - \overline{T})^2} \sqrt{\sum_k \sum_l (I(x+k,y+l) - \overline{I(x,y)})^2}}$$
Normalized cross-correlation – in picture
Normalized cross-correlation – in picture

- well, definitely not convex
- but the discriminability looks promising
- very efficient in computation, see [3]².

²check also normxcorr2 in Matlab
Sum of squared differences

$$ssd(x, y) = \sum_k \sum_l (T(k, l) - I(x + k, y + l))^2$$
Sum of absolute differences

\[ \text{sad}(x, y) = \sum_k \sum_l |T(k, l) - I(x + k, y + l)| \]
SAD for the door part
SAD for the door part – truncated

Differences greater than 20 intensity levels are counted as 20.
Normalized cross-correlation: how it works

live demo for various patches
Normalized cross-correlation: tracking

video
Normalized cross-correlation: tracking

- What went wrong?
- Why did it fail?

Suggestions for improvement?
Tracking as an optimization problem

- finding extrema of a criterial function . . .
Tracking as an optimization problem

- finding extrema of a criterial function . . .
- . . . sounds like an optimization problem

Kanade–Lucas–Tomasi (KLT) tracker

- Iteratively minimizes sum of square differences.
- It is a Gauss-Newton gradient algorithm.
Importance in Computer Vision

- Firstly published in 1981 as an image registration method [4].
- Improved many times, most importantly by Carlo Tomasi [5, 6]
- Free implementation(s) available\(^3\). Also part of the OpenCV library\(^4\).
- After more than two decades, a project\(^5\) at CMU dedicated to this single algorithm and results published in a premium journal [1].
- Part of plethora computer vision algorithms.

Our explanation follows mainly the paper [1]. It is a good reading for those who are also interested in alternative solutions.

\(^3\)http://www.ces.clemson.edu/~stb/klt/
\(^4\)http://opencv.org/
\(^5\)Lucas-Kanade 20 Years On https://www.ri.cmu.edu/research_project_detail.html?project_id=515&menu_id=261
Original Lucas-Kanade algorithm I

**Goal** is to align a template image $T(x)$ to an input image $I(x)$. $x$ column vector containing image coordinates $[x, y]^\top$. The $I(x)$ could be also a small subwindow within an image.

Set of allowable warps $W(x; p)$, where $p$ is a vector of parameters. For translations

$$W(x; p) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$$

$W(x; p)$ can be arbitrarily complex

The best **alignment**, $p^*$, minimizes image dissimilarity

$$\sum_x [I(W(x; p)) - T(x)]^2$$
Original Lucas-Kanade algorithm II

\[ \sum_x [I(W(x; p)) - T(x)]^2 \]

\( I(W(x; p)) \) is nonlinear! The warp \( W(x; p) \) may be linear but the pixels value are, in general, non-linear. In fact, they are essentially unrelated to \( x \).

Linearization of the image: It is assumed that some \( p \) is known and best increment \( \Delta p \) is sought. The modified problem

\[ \sum_x [I(W(x; p + \Delta p)) - T(x)]^2 \]

is solved with respect to \( \Delta p \). When found then \( p \) gets updated

\[ p \leftarrow p + \Delta p \]
Original Lucas-Kanade algorithm III

\[ \sum_x [I(W(x; p + Δp)) - T(x)]^2 \]

linearization by performing first order Taylor expansion\(^6\)

\[ \sum_x [I(W(x; p)) + \nabla I^\top \frac{∂W}{∂p} Δp - T(x)]^2 \]

\[ \nabla I^\top = \left[ \frac{∂I}{∂x}, \frac{∂I}{∂y} \right] \] is the gradient image computed at \( W(x; p) \). The term \( \frac{∂W}{∂p} \) is the Jacobian of the warp.

\(^6\)Detailed explanation on the blackboard.
Differentiate $\sum_x [I(W(x; p)) + \nabla I^\top \frac{\partial W}{\partial p} \Delta p - T(x)]^2$ with respect to $\Delta p$. 
Differentiate $\sum_x [I(W(x; p)) + \nabla I^\top \frac{\partial W}{\partial p} \Delta p - T(x)]^2$ with respect to $\Delta p$ setting equality to zero yields

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

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

setting equality to zero yields

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

where $H$ is (Gauss-Newton) approximation of Hessian matrix.
The Lucas-Kanade algorithm—Summary

Iterate:

1. Warp $I$ with $W(x; p)$
2. Warp the gradient $\nabla I^\top$ with $W(x; p)$
3. Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at $(x; p)$ and compute the steepest descent image $\nabla I^\top \frac{\partial W}{\partial p}$
4. Compute the $H = \sum_x \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]^\top \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]$
5. Compute $\Delta p = H^{-1} \sum_x \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]^\top [T(x) - I(W(x; p))]$
6. Update the parameters $p \leftarrow p + \Delta p$

until $||\Delta p|| \leq \epsilon$
Example of convergence

video
Example of convergence

Convergence video: Initial state is within the basin of attraction
Example of divergence

Divergence video: Initial state is outside the basin of attraction
Example – on-line demo

Let play and see . . .
What are good features (windows) to track?
What are good features (windows) to track?

How to select good templates $T(x)$ for image registration, object tracking.

\[
\Delta p = H^{-1} \sum_x \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]^\top [T(x) - I(W(x; p))] 
\]
What are good features (windows) to track?

How to select good templates $T(x)$ for image registration, object tracking.

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

where $H$ is the matrix

$$H = \sum_x \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]^\top \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]$$
What are good features (windows) to track?

How to select good templates $T(x)$ for image registration, object tracking.

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

where $H$ is the matrix

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

The stability of the iteration is mainly influenced by the inverse of Hessian. We can study its eigenvalues. Consequently, the criterion of a good feature window is $\min(\lambda_1, \lambda_2) > \lambda_{\text{min}}$ (texturedness).
What are good features for translations?

Consider translation $W(x; p) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}$. The Jacobian is then

$$\frac{\partial W}{\partial p} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$H = \sum_x \left( \nabla I^\top \frac{\partial W}{\partial p} \right)^\top \left[ \nabla I^\top \frac{\partial W}{\partial p} \right]$$

$$= \sum_x \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x}, \frac{\partial I}{\partial x}, \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \sum_x \begin{bmatrix} (\frac{\partial I}{\partial x})^2 & \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} & (\frac{\partial I}{\partial y})^2 \end{bmatrix}$$

The image windows with varying derivatives in both directions. Homeogeneous areas are clearly not suitable. Texture oriented mostly in one direction only would cause instability for this translation.
What are the good points for translations?

The matrix

$$H = \sum_x \begin{bmatrix} (\frac{\partial I}{\partial x})^2 & \frac{\partial I \partial I}{\partial x \partial y} \\ \frac{\partial I \partial I}{\partial x \partial y} & (\frac{\partial I}{\partial y})^2 \end{bmatrix}$$

Should have large eigenvalues. We have seen the matrix already, where?
What are the good points for translations?

The matrix

\[ H = \sum_x \begin{bmatrix} \left( \frac{\partial I}{\partial x} \right)^2 & \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} & \left( \frac{\partial I}{\partial y} \right)^2 \end{bmatrix} \]

Should have large eigenvalues. We have seen the matrix already, where?

Harris corner detector [2]! The matrix is sometimes called Harris matrix.
Experiments - no occlusions
Experiments - occlusions
Experiments - occlusions with dissimilarity
Experiments - object motion
Experiments – door tracking
Experiments – door tracking – smoothed
Comparison of ncc vs KLT tracking
References


End
criterial function ncc

100 200 300 400 500 600 700

50 100 150 200 250 300 350 400 450 500 550

−0.6 −0.4 −0.2 0 0.2 0.4 0.6 0.8
criterial function ssd

The image shows a heatmap with values ranging from 0 to $10^7$. The x-axis and y-axis values are not explicitly labeled, but they range from 100 to 700. The heatmap indicates varying levels of magnitude across these values, with the color scale on the right side ranging from dark blue (0) to red ($10^7$). The pattern and distribution of the colors suggest a specific function or data pattern that the ssd (sum of squares difference) criterion is applied to. A green cross appears in the center of the heatmap, potentially indicating a point of interest or reference within the data set.