Lukas-Kanade Tracker[1]

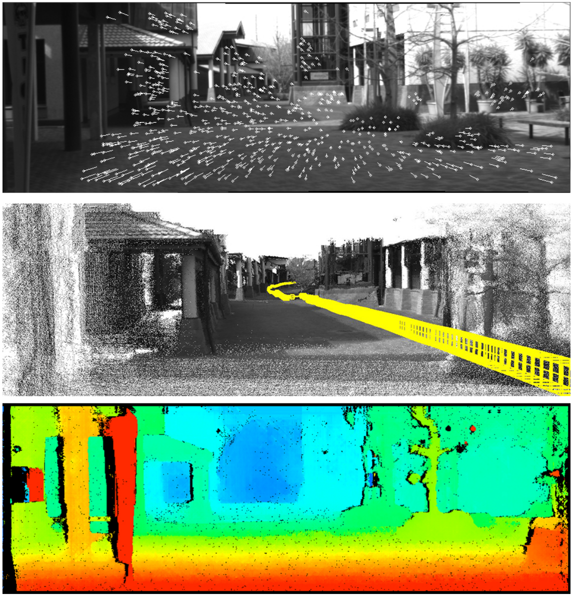
Lecture 23

See Sections 9.3.1 and 9.3.2 in
Reinhard Klette: Concise Computer Vision
Springer-Verlag, London, 2014

[1]See last slide for copyright information.

Agenda

Example



Caption

Top: Tracked feature points in a frame of a stereo video sequence recorded in a car

Middle: Tracked feature points are used for calculating the motion of the car; this allows to map 3D points provided by stereo vision into a uniform 3D world coordinate system

Bottom: Stereo matcher `iSGM` has been used for the shown example (example of a disparity map for the recorded sequence).

Example of an Application Scenario

A car, which is called the *ego-vehicle* because it is the reference vehicle where the considered system is working in, in distinction to "other" vehicles in a scene

This ego-vehicle is equipped with a stereo vision system and it drives through a street, providing reconstructed 3D clouds of points for each stereo frame at time t

After understanding the motion of the ego-vehicle, these 3D clouds of points can be mapped into a uniform 3D world coordinate system supporting 3D surface modeling of the road sides

For understanding the motion of the ego-vehicle, we track detected features from Frame t to Frame $t+1$, being the input for a program calculating the *ego-motion* of the car

Tracking is a Sparse Correspondence Problem

Binocular stereo
Point or feature correspondence is calculated between images taken at the same time; the correspondence search is within an epipolar line; thus, stereo matching is a *1D correspondence problem*

Dense Motion (i.e. optic flow) Analysis
Point or feature correspondence is calculated between images taken at subsequent time slots; movements of pixels not constrained to be along one straight line; dense motion analysis is a *2D correspondence problem*

Feature Tracking
A sparse 2D correspondence problem

Tracking and Updating of Features

Theoretically, its solution could also be used for solving stereo or dense motion analysis

But there are different strategies for solving a dense or a sparse correspondence problem

In sparse correspondence search we cannot utilize a smoothness term, and need to focus more at first on achieving accuracy based on the data term only

We can use global consistency of tracked feature point patterns for stabilizing the result

Tracking with Understanding 3D Changes

Pair of 3D points $P_t = (X_t, Y_t, Z_t)$ and $P_{t+1} = (X_{t+1}, Y_{t+1}, Z_{t+1})$, projected at times t and $t+1$ into $p_t = (x_t, y_t, f)$ and $p_{t+1} = (x_{t+1}, y_{t+1}, f)$, respectively, when recording a video sequence

Z-ratio

$$\psi_Z = \frac{Z_{t+1}}{Z_t}$$

We can derive X- and Y-ratios

$$\begin{aligned} \psi_X &= \frac{X_{t+1}}{X_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{x_{t+1}}{x_t} = \psi_Z \frac{x_{t+1}}{x_t} \\ \psi_Y &= \frac{Y_{t+1}}{Y_t} = \frac{Z_{t+1}}{Z_t} \cdot \frac{y_{t+1}}{y_t} = \psi_Z \frac{y_{t+1}}{y_t} \end{aligned}$$

Update Equation

$$\begin{bmatrix} X_{t+1} \\ Y_{t+1} \\ Z_{t+1} \end{bmatrix} = \begin{bmatrix} \psi_X & 0 & 0 \\ 0 & \psi_Y & 0 \\ 0 & 0 & \psi_Z \end{bmatrix} \cdot \begin{bmatrix} X_t \\ Y_t \\ Z_t \end{bmatrix}$$

Knowing ψ_Z and ratios $\frac{x_{t+1}}{x_t}$ and $\frac{y_{t+1}}{y_t}$ allows us to update the position of point P_t into P_{t+1}

Assuming that P_t and P_{t+1} are positions of a 3D point P, from time t to time $t+1$, we only have to

1. decide on a technique to track points from t to $t+1$
2. estimate ψ_Z

Initial Position and Z-Ratios

If an initial position P_0 of a tracked point P is known then we may identify its 3D position at subsequent time slots

Without having an initial position, we only have a 3D direction P_t to P_{t+1}, but not its 3D position

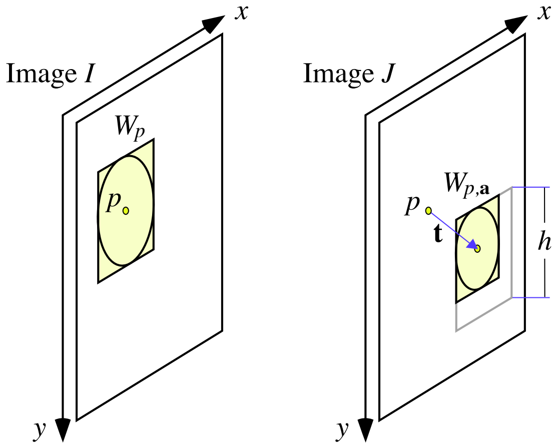
Stereo vision is the general solution for estimating Z-values or (just) ratios ψ_Z

We can also estimate ψ_Z in a monocular sequence from scale-space results

Now: how to track points from t to $t+1$?

Agenda

Lucas-Kanade Tracker



Template or base window W_p in base image I compared with a match window $W_{p,\mathbf{a}}$ in match image J

Sketch

Shown case: dissimilarity vector $\mathbf{a}$ is a translation $\mathbf{t}$ and a scaling of height h into a smaller height

Figure indicates that a disk of influence is contained in W_p

Pixel location p in J is the same as in I; it defines the start of the translation

Lucas-Kanade Tracker

Match *template* W_p, being a $(2k+1) \times (2k+1)$ window around keypoint $p = (x, y)$ in a base image I, with windows $W_{p,\mathbf{a}}$ in a match image J

Method should be general enough to allow for translation, scaling, rotation and so forth between base window W_p and match window $W_{p,\mathbf{a}}$ in J

Vector $\mathbf{a}$ parametrizes the transform from p into a new center pixel, and also the transformation of window W into a new shape

Newton-Raphson Iteration

Task: Calculate a zero of a smooth unary function $\phi(x)$, for $x \in [a, b]$, provided that we have $\phi(a)\phi(b) < 0$

Inputs are the two reals a and b

We also have a way to calculate $\phi(x)$ and the derivative $\phi'(x)$ (e.g. approximated by difference quotients), for any $x \in [a, b]$

Calculate a value $c \in [a, b]$ as an approximate zero of ϕ:

```
1: Let $c \in [a, b]$ be an initial guess for a zero.
2: **while** STOP CRITERION $=$ false **do**
3:     Replace $c$ by $c - \frac{\phi(c)}{\phi'(c)}$
4: **end while**
```

Derivative $\phi'(c)$ is assumed to be non-zero; if ϕ has a derivative of constant sign in $[a, b]$ then there is just one zero in $[a, b]$

Comments

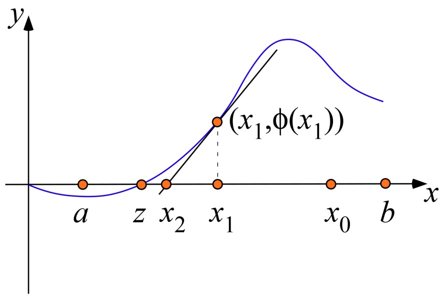
Initial value of c can be specified by (say) a small number of binary-search steps for reducing the run-time of the actual Newton-Raphson iteration

A small $\varepsilon > 0$ is used for specifying the STOP CRITERION "$|\phi(c)| \leq \varepsilon$"

Method converges in general only if c is "sufficiently close" to the zero z

If $\phi''(x)$ has a constant sign in $[a, b]$, then we have the following: if $\phi(b)$ has the same sign as $\phi''(x)$ then initial value $c = b$ gives convergence to z, otherwise chose initial value $c = a$

Figure



A smooth function $\phi(x)$ on an interval $[a, b]$ with $\phi(a)\phi(b) < 0$

Assume that we start with $c = x_1$

Tangent at $(x_1, \phi(x_1))$ intersects x-axis at x_2 and defined by

$$x_2 = x_1 - \frac{\phi(x_1)}{\phi'(x_1)}$$

Have $\phi'(x_1) \neq 0$. Now continue with $c = x_2$ and new tangent, etc.

Convergence and Valleys

For initial value x_1, sequence $x_2, x_3, \ldots$ converges to zero z

If start at $c = x_0$ then the algorithm would fail

Note that $\phi''(x)$ does not have a constant sign in $[a, b]$

We need to start in the "same valley" where z is located

We search for the zero in the direction of the (steepest) decent

If we do not start in the "same valley" then we cannot cross the "hill" in between

Following the Newton-Raphson Iteration.

Lucas-Kanade tracker uses approximate gradients which are robust against variations in intensities

For window matching, an error function E is defined based on an LSE optimization criterion

Agenda

Translation

Simplest case: only a translation $\mathbf{t} = [t.x, t.y]^\top$ such that $J(x + t.x + i, y + t.y + j) \approx I(x + i, y + j)$, for all i, j, with $-k \leq i, j \leq k$, defining relative locations in template W_p

Simplifying notation: assume that $p = (x, y) = (0, 0)$, and we use W or $W_\mathbf{a}$ instead of W_p or $W_{p,\mathbf{a}}$, respectively

Case of translation-only: approximate a zero (i.e. a minimum) of the error function

$$E(\mathbf{t}) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \left[J(t.x + i, t.y + j) - I(W(i,j))\right]^2$$

where $\mathbf{t} = [t.x, t, y]^\top$ and $W(i,j) = (i,j)$

Goal for General Warps

Tracker not just for translations but for general *warps* defined by an affine transform, with a vector **a** parametrizing the transform

Let $J(W_{\mathbf{a}}(q))$ be the value at that point $W_{\mathbf{a}}(q)$ in J which results from warping pixel location $q = (i, j)$, with $-k \leq i, j \leq k$, according to parameter vector **a**

Warping will not map a pixel location onto a pixel location, thus we also apply some kind of interpolation for defining $J(W_{\mathbf{a}}(q))$

Translation with $\mathbf{a} = [t.x, t.y]^\top$: for $q = (i, j)$ we have $W_{\mathbf{a}}(q) = (t.x, t.y) + q$ and $J(W_{\mathbf{a}}(q)) = J(t.x + i, t.y + j)$

General case: calculate *dissimilarity vector* **a** which minimizes error function

$$E(\mathbf{a}) = \sum_q [J(W_{\mathbf{a}}(q)) - I(W(q))]^2$$

Iterative Steepest-Ascent Algorithm

Assume: we are at a parameter vector $\mathbf{a} = [a_1, \ldots, a_n]^\top$

Similarly to mean-shift algorithm for image segmentation, calculate (as partial step) shift $m_\mathbf{a} = [m_1, \ldots, m_n]^\top$ which minimizes

$$E(\mathbf{a} + m_\mathbf{a}) = \sum_q [J(W_{\mathbf{a}+m_\mathbf{a}}(q)) - I(W(q))]^2$$

Solving this LSE optimization problem:

Consider Taylor expansion (analog to deriving the Horn-Schunck constraint) of $J(W_\mathbf{a}(q))$ with respect to dissimilarity vector $\mathbf{a}$ and a minor shift $m_\mathbf{a}$

$$J(W_{\mathbf{a}+m_\mathbf{a}}(q)) = J(W_\mathbf{a}(q)) + m_\mathbf{a}^\top \cdot \mathbf{grad}\, J \cdot \frac{\partial W_\mathbf{a}}{\partial \mathbf{a}} + e$$

Assume $e = 0$, thus linearity of values of image J in the neighborhood of pixel location $W_\mathbf{a}(q)$

LSE Optimization Problem

Second term on the right-hand side is a scalar: product of shift vector $m_{\mathbf{a}}$, derivative **grad** J of the outer function (i.e. the usual image gradient), and the derivative of the inner function

Window function W defines a point with x- and y-coordinates; derivative of W with respect to locations identified by **a**:

$$\frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}(q) = \begin{bmatrix} \frac{\partial W_{\mathbf{a}}(q).x}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).x}{\partial y} \\ \frac{\partial W_{\mathbf{a}}(q).y}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).y}{\partial y} \end{bmatrix}$$

This is the *Jacobian matrix* of the warp; minimization problem now:

$$\sum_q \left[J(W_{\mathbf{a}}(q)) + m_{\mathbf{a}}^{\top} \cdot \mathbf{grad}\, J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \; - \; I(W(q)) \right]^2$$

Follow standard LSE optimization for calculating optimum shift $m_{\mathbf{a}}$

LSE Procedure

1. Calculate the derivative of this sum with respect to shift $m_{\mathbf{a}}$
2. Set this equal to zero
3. Obtain the equation (with 2×1 zero-vector $\mathbf{0}$)

$$2 \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^\top \left[J(W_{\mathbf{a}}(q)) + m_{\mathbf{a}}^\top \cdot \mathbf{grad}\ J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \quad - \quad I(W(q))\right] = \mathbf{0}$$

2×2 Hessian matrix

$$H = \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^\top \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]$$

Solution defines optimum shift vector $m_{\mathbf{a}}$

$$m_{\mathbf{a}}^\top = H^{-1} \sum_q \left[\mathbf{grad}\ J\ \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^\top [I(W(q)) - J(W_{\mathbf{a}}(q))]$$

from given parameter vector $\mathbf{a}$ to updated vector $\mathbf{a} + m_{\mathbf{a}}$

Analogy to the Newton-Raphson Iteration

1. Start with an initial dissimilarity vector $\mathbf{a}$
2. New vectors $\mathbf{a} + m_{\mathbf{a}}$ are calculated in iterations
3. Follow the steepest ascent

Possible stop criteria

1. Error value or length of shift vector $m_{\mathbf{a}}$ is below a given $\varepsilon > 0$
2. A predefined maximum of iterations

Example: Translation Case

Only translation $\mathbf{a}$ with $W_{\mathbf{a}}(q) = [t.x + i, t_y + j]^\top$, for $q = (i, j)$

Jacobian matrix

$$\frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}(q, \mathbf{a}) = \begin{bmatrix} \frac{\partial W_{\mathbf{a}}(q).x}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).x}{\partial y} \\ \frac{\partial W_{\mathbf{a}}(q).y}{\partial x} & \frac{\partial W_{\mathbf{a}}(q).y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Hessian matrix (approximated by products of first-order derivatives)

$$H = \sum_q \left[\mathbf{grad}\ J \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right]^\top \left[\mathbf{grad}\ J \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}}\right] = \sum_q \begin{bmatrix} \left(\frac{\partial J}{\partial x}\right)^2 & \frac{\partial J^2}{\partial x \partial y} \\ \frac{\partial J^2}{\partial x \partial y} & \left(\frac{\partial J}{\partial y}\right)^2 \end{bmatrix}$$

Steepest ascent

$$\mathbf{grad}\ J \cdot \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} = \mathbf{grad}\ J$$

and

$$I(W(q)) - J(W_{\mathbf{a}}(q)) = I(W(q)) - J(q + \mathbf{a})$$

Altogether (for Translation)

$$
\begin{aligned}
m_{\mathbf{a}}^{\top} &= H^{-1} \sum_{q} \left[\mathbf{grad}\ J \, \frac{\partial W_{\mathbf{a}}}{\partial \mathbf{a}} \right]^{\top} \left[I(W(q)) - J(W_{\mathbf{a}}(q)) \right] \\
&= \left[\sum_{q} \begin{bmatrix} \left(\frac{\partial J}{\partial x}\right)^2 & \frac{\partial J^2}{\partial x \partial y} \\ \frac{\partial J^2}{\partial x \partial y} & \left(\frac{\partial J}{\partial y}\right)^2 \end{bmatrix} \right]^{-1} \sum_{q} [\mathbf{grad}\ J]^{\top} \left[I(W(q)) - J(q + \mathbf{a}) \right]
\end{aligned}
$$

1. Approximate derivatives in image J around the current pixel locations in window W
2. This defines the Hessian and the gradient vector
3. Then a sum of differences for identifying the shift vector $m_{\mathbf{a}}$

Lucas-Kanade Algorithm

Given is an image I, its gradient image **grad** I, and a local template W (i.e. a window) containing (e.g.) the disk of influence of a keypoint

```
1: Let $\mathbf{a}$ be an initial guess for a dissimilarity vector
2: while STOP CRITERION = false do
3:     For the given vector $\mathbf{a}$, compute the optimum shift $m_{\mathbf{a}}$ as
       defined above
4:     Let $\mathbf{a} = \mathbf{a} + m_{\mathbf{a}}$
5: end while
```

Line 3

Line 3 in the algorithm requires calculations for all pixels q defined by template W; basically the main steps:

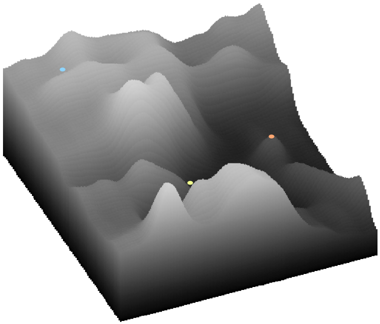
1. Warp W in I into $W_{\mathbf{a}}(q)$ in J
2. Calculate the Jacobian matrix and its product with $\mathbf{grad}\ J$
3. Compute the Hessian matrix

The algorithm performs magnitudes faster than an exhaustive search algorithm for an optimized vector $\mathbf{a}$

Program for Lucas-Kanade algorithm available in `OpenCV`

Dents and Hills

Assume that error values are defined on the plane, and for different values of **a** they describe a "hilly terrain", with local minima, possibly a uniquely defined global minimum, local maxima, and possibly a uniquely defined global maximum



Blue point "cannot climb" to global maximum; red point is already at local maximum; yellow dot can iterate to global peak

Drift

There is also the possibility of a *drift*

The individual local calculation can be accurate, but the composition of several local moves may result in significant errors after some time, mainly due to the discrete nature of the data

Copyright Information