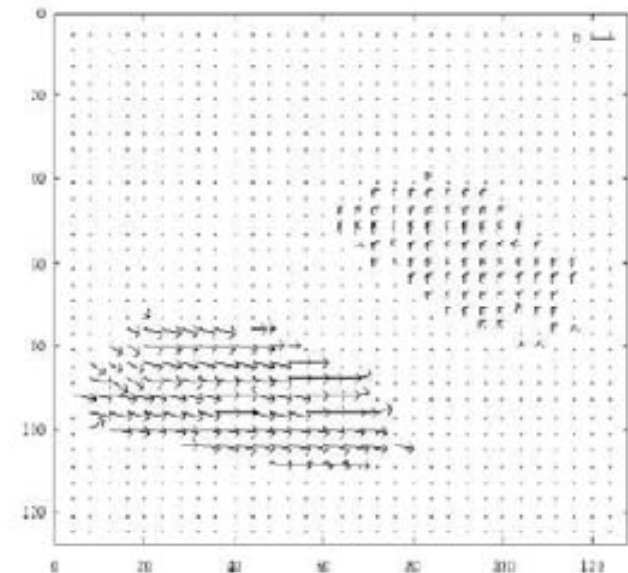


Optical Flow, KLT Feature Tracker

Motion in Computer Vision

Motion

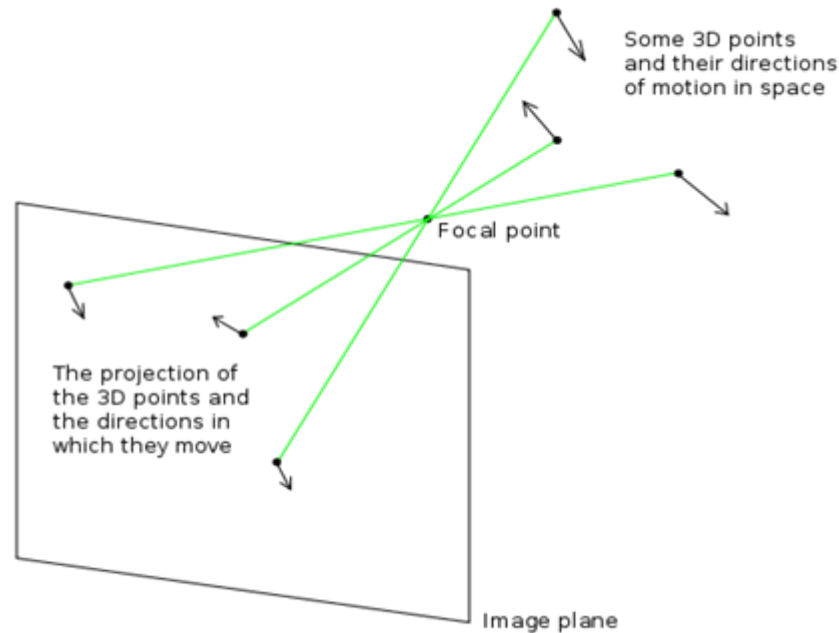
- Structure from motion
- Detection/segmentation with direction



[1]

Motion Field v.s. Optical Flow [2], [3]

Motion Field: an ideal representation of 3D motion as it is projected onto a camera image.



Optical Flow: the approximation (or estimate) of the motion field which can be computed from time-varying image sequences. Under the simplifying assumptions of 1) Lambertian surface, 2) pointwise light source at infinity, and 3) no photometric distortion.

Motion Field [2]

- An ideal representation of 3D motion as it is projected onto a camera image.
- The time derivative of the image position of all image points given that they correspond to fixed 3D points. “field : position \rightarrow vector”

- The motion field \mathbf{v} is defined as
$$\mathbf{v} = f \frac{Z\mathbf{V} - V_z\mathbf{P}}{Z^2}$$

where $\mathbf{V} = -\mathbf{T} - \boldsymbol{\omega} \times \mathbf{P}$

\mathbf{P} is a point in the scene where Z is the distance to that scene point.

\mathbf{V} is the relative motion between the camera and the scene,

\mathbf{T} is the translational component of the motion,

and $\boldsymbol{\omega}$ is the angular velocity of the motion.

Motion Field [2]

3D point $P (X,Y,Z)$ and 2D point $\mathbf{p} (x,y)$, focal length f

$$\mathbf{p} = f \frac{\mathbf{P}}{Z} \quad \begin{cases} x = f \frac{X}{Z} \\ y = f \frac{Y}{Z} \end{cases} \quad (1)$$

Motion field \mathbf{v} can be obtained by taking the time derivative of (1)

$$\begin{cases} v_x = f \left(\frac{V_x}{Z} - X \frac{V_z}{Z^2} \right) \\ v_y = f \left(\frac{V_y}{Z} - Y \frac{V_z}{Z^2} \right) \end{cases} \quad (2)$$

Motion Field [2]

The motion of 3D point P, V is defined as flow

$$\mathbf{V} = -\mathbf{T} - \boldsymbol{\omega} \times \mathbf{P} \quad \begin{cases} V_X = -T_X - \omega_Y Z + \omega_Z Y \\ V_Y = -T_Y - \omega_Z X + \omega_X Z \\ V_Z = -T_Z - \omega_X Y + \omega_Y X \end{cases} \quad (3)$$

By substituting (3) into (2), the basic equations of the motion field is acquired

$$\begin{cases} v_x = \frac{T_Z x - T_X f}{Z} - \omega_Y f + \omega_Z y + \frac{\omega_X xy}{f} - \frac{\omega_Y x^2}{f} \\ v_y = \frac{T_Z y - T_Y f}{Z} + \omega_X f - \omega_Z x - \frac{\omega_Y xy}{f} + \frac{\omega_X y^2}{f} \end{cases} \quad (4)$$

Motion Field [2]

The motion field is the sum of two components, one of which depends on translation only, the other on rotation only.

$$\begin{cases} v_x = \frac{T_Z x - T_X f}{Z} - \omega_Y f + \omega_Z y + \frac{\omega_X xy}{f} - \frac{\omega_Y x^2}{f} \\ v_y = \frac{T_Z y - T_Y f}{Z} + \omega_X f - \omega_Z x - \frac{\omega_Y xy}{f} + \frac{\omega_X y^2}{f} \end{cases} \quad (4)$$

Translational
components

Rotational
components

$$\begin{cases} v_x = \frac{T_Z x - T_X f}{Z} \\ v_y = \frac{T_Z y - T_Y f}{Z} \end{cases} \quad (5)$$

$$\begin{cases} v_x = -\omega_Y f + \omega_Z y + \frac{\omega_X xy}{f} - \frac{\omega_Y x^2}{f} \\ v_y = +\omega_X f - \omega_Z x - \frac{\omega_Y xy}{f} + \frac{\omega_X y^2}{f} \end{cases} \quad (6)$$

Motion Field: Pure Translation [2]

If there is no rotational motion, the resulting motion field has a peculiar spatial structure.

If (5) is regarded as a function of 2D point position,

$$\begin{cases} v_x = \frac{T_Z x - T_X f}{Z} \\ v_y = \frac{T_Z y - T_Y f}{Z} \end{cases} \quad (5) \quad \Rightarrow \quad \begin{cases} v_x = \frac{T_Z}{Z} \left(x - f \frac{T_X}{T_Z} \right) \\ v_y = \frac{T_Z}{Z} \left(y - f \frac{T_Y}{T_Z} \right) \end{cases}$$

If (x_0, y_0) is defined as in (6)

$$\begin{cases} x_0 = f \frac{T_X}{T_Z} \\ y_0 = f \frac{T_Y}{T_Z} \end{cases} \quad (6)$$

$$\begin{cases} v_x = \frac{T_Z}{Z} (x - x_0) \\ v_y = \frac{T_Z}{Z} (y - y_0) \end{cases} \quad (7)$$

Motion Field: Pure Translation [2]

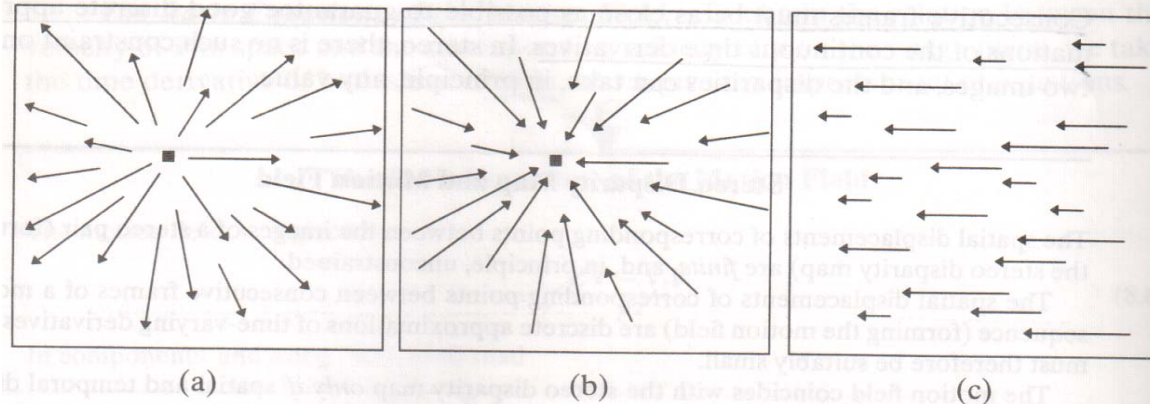


Figure 8.4 The three types of motion field generated by translational motion. The filled square marks the instantaneous epipole.

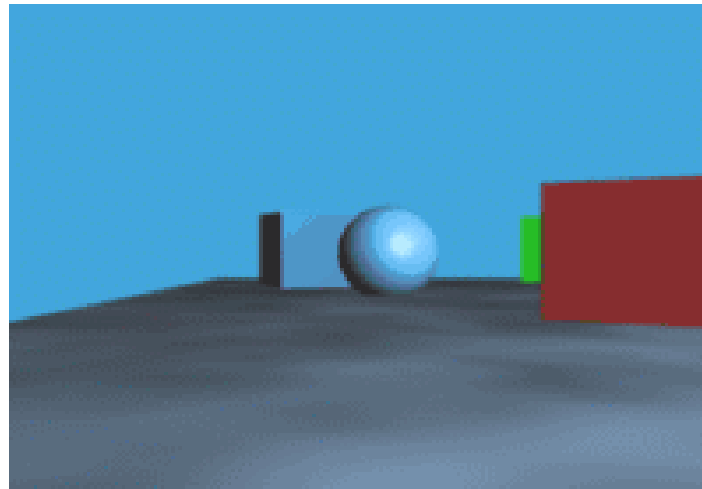
Equation (7) say that the motion field of a pure translation is radial. In particular, if $T_z < 0$, the vectors point away from $\mathbf{p}_0 (x_0, y_0)$, which is called the **focus of expansion (FOE)**. If $T_z > 0$, the motion field vectors point towards \mathbf{p}_0 , which is called the **focus of contraction**. If $T_z = 0$, from (5), all the motion field vectors are parallel.

$$\begin{cases} v_x = \frac{T_z x - T_x f}{Z} \\ v_y = \frac{T_z y - T_y f}{Z} \end{cases} \quad (5) \quad \longrightarrow \quad \begin{cases} v_x = -f \frac{T_x}{Z} \\ v_y = -f \frac{T_y}{Z} \end{cases} \quad (8)$$

Motion Field: Motion Parallax [6]

Equation (8) say that their lengths are inversely proportional to the depth of the corresponding 3D points.

$$\begin{cases} v_x = -f \frac{T_x}{Z} \\ v_y = -f \frac{T_y}{Z} \end{cases} \quad (8)$$



<http://upload.wikimedia.org/wikipedia/commons/a/ab/Parallax.gif>

This animation is an example of parallax. As the viewpoint moves side to side, the objects in the distance appear to move more slowly than the objects close to the camera [6].

Motion Field: Motion Parallax [2]

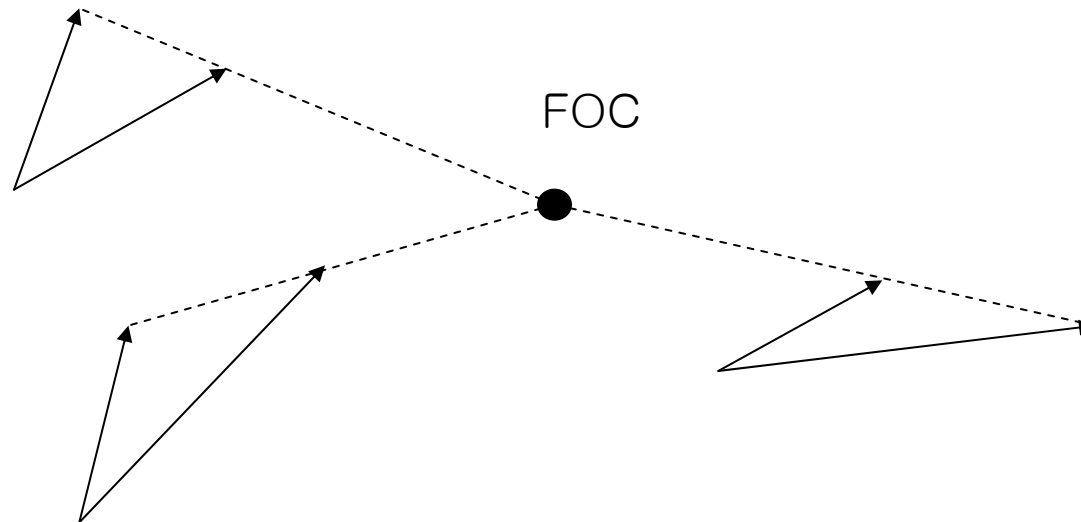
If two 3D points are projected into one image point, that is coincident, rotational component will be the same. Notice that the motion vector V is about camera motion.

$$\begin{cases} v_x = \frac{T_Z x - T_X f}{Z} - \omega_Y f + \omega_Z y + \frac{\omega_X xy}{f} - \frac{\omega_Y x^2}{f} \\ v_y = \frac{T_Z y - T_Y f}{Z} + \omega_X f - \omega_Z x - \frac{\omega_Y xy}{f} + \frac{\omega_X y^2}{f} \end{cases} \quad (4)$$

Motion Field: Motion Parallax [2]

The difference of two points' motion field will be related with translation components. And, they will be radial w.r.t FOE or FOC.

$$\begin{cases} \Delta v_x = (T_z x - T_x f) \left(\frac{1}{Z_1} - \frac{1}{Z_2} \right) = (x - x_0) T_z \left(\frac{1}{Z_1} - \frac{1}{Z_2} \right) \\ \Delta v_y = (T_z y - T_y f) \left(\frac{1}{Z_1} - \frac{1}{Z_2} \right) = (y - y_0) T_z \left(\frac{1}{Z_1} - \frac{1}{Z_2} \right) \end{cases}$$



Motion Field: Motion Parallax [2]

Motion Parallax

The relative motion field of two instantaneously coincident points:

1. Does not depend on the rotational component of motion
2. Points towards (away from) the point p_0 , the vanishing point of the translation direction.

Motion Field: Pure Rotation w.r.t Y-axis [7]

If there is no translation motion and rotation w.r.t x- and z- axis, from (4)

$$\begin{cases} v_x = -\omega_Y f - \frac{\omega_Y x^2}{f} \\ v_y = \frac{\omega_Y xy}{f} \end{cases}$$

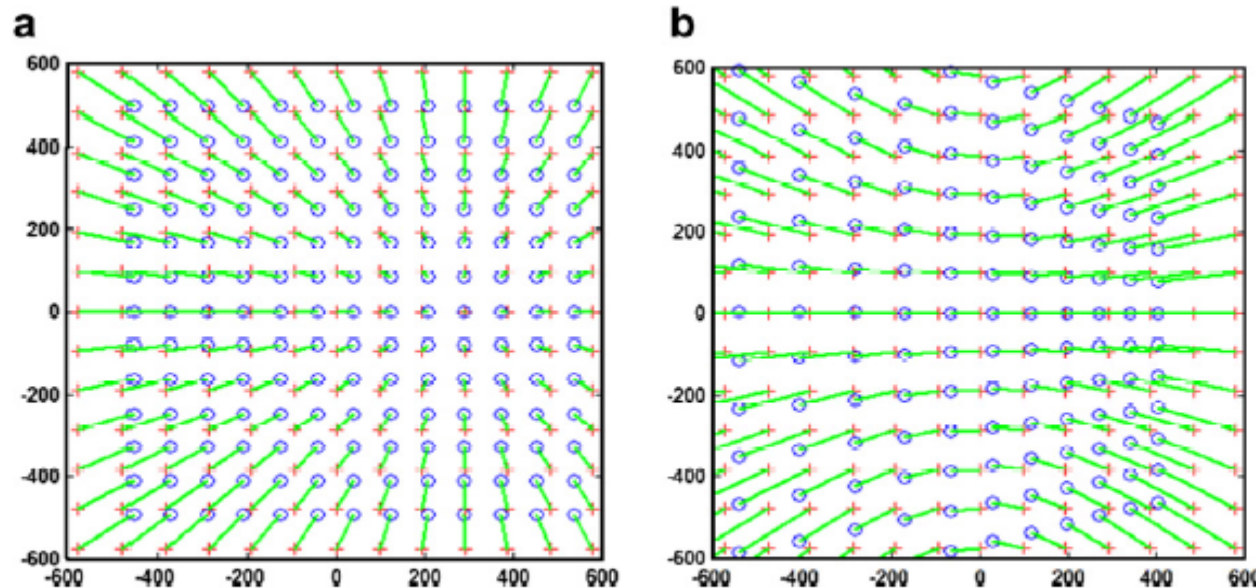
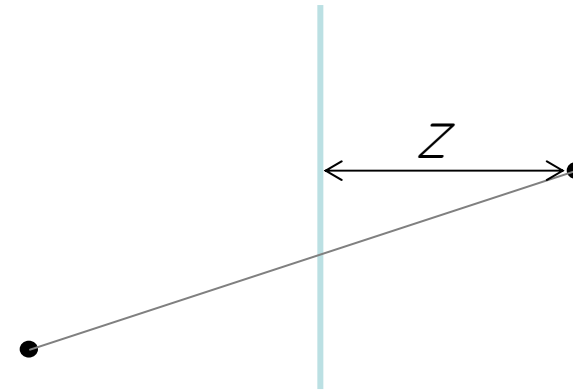
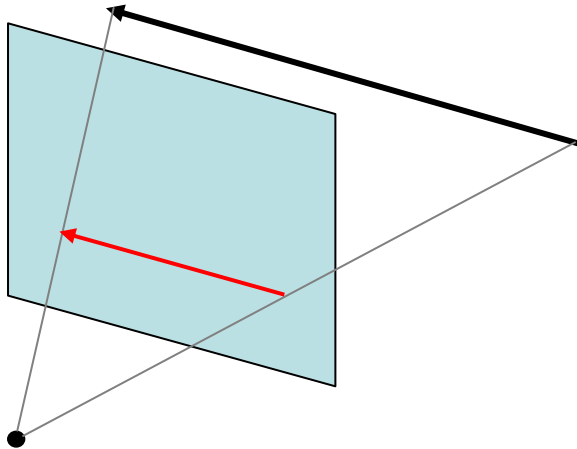


Fig. 2. (a) Motion field of a pure translation (T_x and T_z). (b) Motion field of a pure rotation (R_y). The crosses and circles represent the feature points in the first and the second images, respectively, and the lines join the matches.

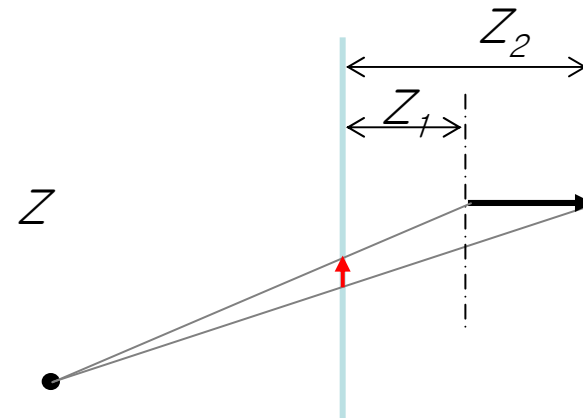
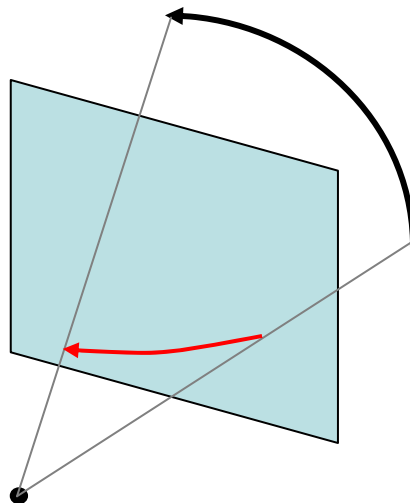
Motion Field: Pure Rotation w.r.t Y-axis [7]

Translational Motion



Distance to the point, Z , is constant.

Rotational Motion



Distance to the point, Z , is changing.
According to Z , y is changing, too.

Estimation of the Optical Flow [4]

For a 2D+t dimensional case (3D or n-D cases are similar) a voxel at location (x, y, t) with intensity $I(x, y, t)$ will have moved by δx , δy and δt between the two image frames, and the following *image constraint equation* can be given:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

Assuming the movement to be small, the image constraint at $I(x, y, t)$ with Taylor series can be developed to get

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t + H.O.T.$$

where H.O.T. means higher order terms, which are small enough to be ignored. From these equations it follows that

$$\frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = 0$$

or

$$\frac{\partial I}{\partial x} \frac{\delta x}{\delta t} + \frac{\partial I}{\partial y} \frac{\delta y}{\delta t} + \frac{\partial I}{\partial t} = 0$$

Estimation of the Optical Flow [4]

which results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

where V_x, V_y are the x and y components of the velocity or optical flow of $I(x,y,t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x,y,t) in the corresponding directions. I_x, I_y and I_t can be written for the derivatives in the following.

Thus:

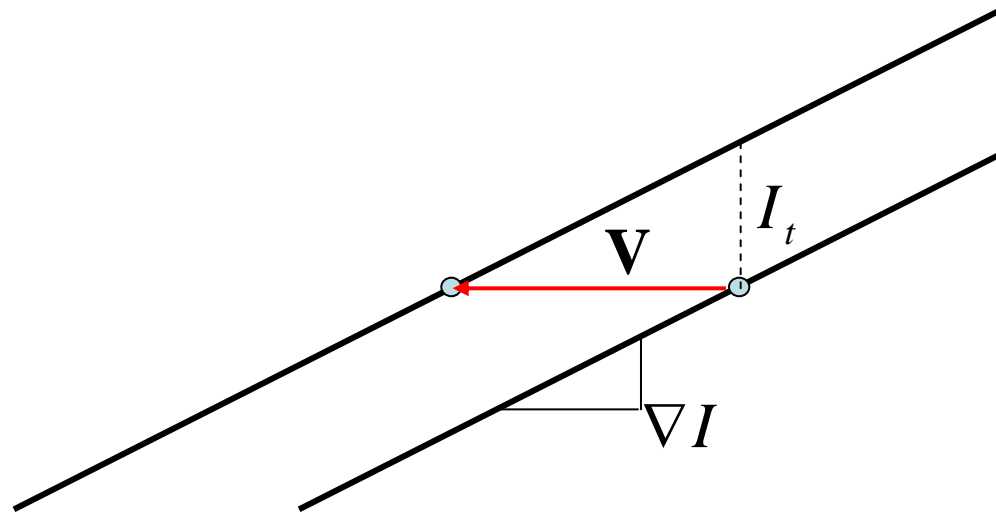
$$I_xV_x + I_yV_y = -I_t$$

or

$$\nabla I^T \cdot \vec{V} = -I_t \quad \text{The Image Brightness Constancy Equation [2]}$$

Estimation of the Optical Flow [4]

$$\nabla I^T \cdot \mathbf{V} = -I_t$$



→ Assumption

The image brightness is continuous and differentiable as many times as needed in both the spatial and temporal domain.

The image brightness can be regarded as a plane in a small area.

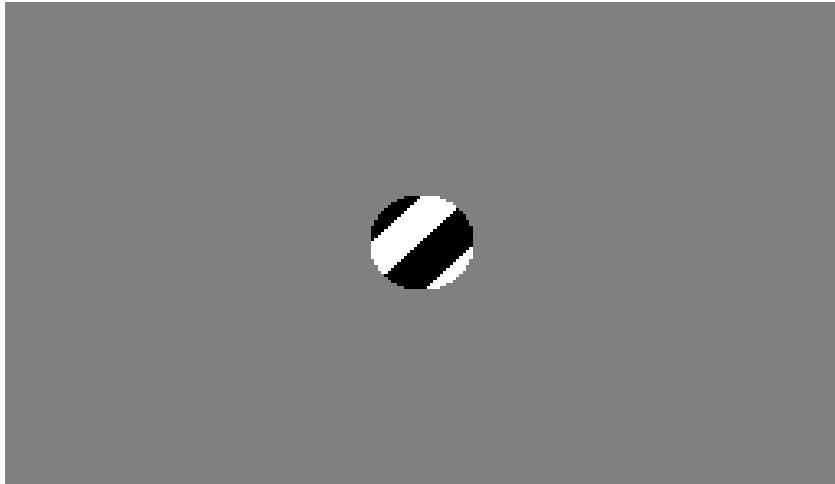
Optical Field: Aperture Problem [2], [4], [9]

$$\nabla I^T \cdot \vec{V} = -I_t$$

This is an equation in two unknowns and cannot be solved as such. This is known as the *aperture problem* of the optical flow algorithms. To find the optical flow another set of equations is needed, given by some additional constraint. All optical flow methods introduce additional conditions for estimating the actual flow.

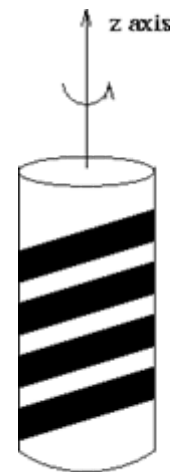
The component of the motion field in the direction orthogonal to the spatial image gradient is not constrained by the image brightness constancy equation. ➔ Given local information can determine component of optical flow vector only in direction of brightness gradient.

Optical Field: Aperture Problem [2], [9]

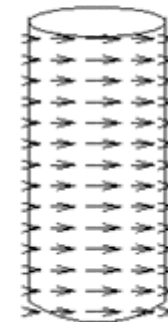


The aperture problem. The [grating](#) appears to be moving down and to the right, [perpendicular](#) to the orientation of the bars. But it could be moving in many other directions, such as only down, or only to the right. It is impossible to determine unless the ends of the bars become visible in the aperture.

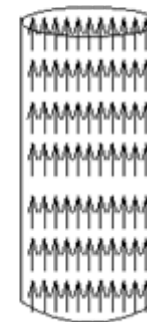
http://upload.wikimedia.org/wikipedia/commons/f/f0/Aperture_problem_animated.gif



Barber's pole



Motion field



Optical flow

Optical Field: Methods for Determining Optical Flow [4]

- Phase correlation – inverse of normalized cross-power spectrum
- Block-based methods – minimizing sum of squared differences or sum of absolute differences, or maximizing normalized cross-correlation
- Differential methods of estimating optical flow, based on partial derivatives of the image signal and/or the sought flow field and higher-order partial derivatives, such as:
 - Lucas–Kanade method – regarding image patches and an affine model for the flow field
 - Horn–Schunck method – optimizing a functional based on residuals from the brightness constancy constraint, and a particular regularization term expressing the expected smoothness of the flow field
 - Buxton–Buxton method – based on a model of the motion of edges in image sequences^[9]
 - Black–Jepson method – coarse optical flow via correlation^[6]
 - General variational methods – a range of modifications/extensions of Horn–Schunck, using other data terms and other smoothness terms.
- Discrete optimization methods – the search space is quantized, and then image matching is addressed through label assignment at every pixel, such that the corresponding deformation minimizes the distance between the source and the target image.^[10] The optimal solution is often recovered through min-cut max-flow algorithms, linear programming or belief propagation methods.

Optical Field: Phase Correlation Method [10]

Given two input images g_a and g_b :

Apply a [window function](#) (e.g., a [Hamming window](#)) on both images to reduce edge effects. Then, calculate the discrete 2D [Fourier transform](#) of both images.

Calculate the [cross-power spectrum](#) by taking the [complex conjugate](#) of the second result, multiplying the [Fourier transforms](#) together elementwise, and normalizing this product elementwise.

$$R = \frac{\mathbf{G}_a \mathbf{G}_b^*}{|\mathbf{G}_a \mathbf{G}_b^*|}$$

Obtain the normalized cross-correlation by applying the inverse Fourier transform.

$$r = \mathcal{F}^{-1}\{R\}$$

[Determine the location of the peak](#) in r (possibly using sub-pixel edge detection).

$$(\Delta x, \Delta y) = \arg \max_{(x,y)} \{r\}$$

Optical Field: Phase Correlation Method [10]

The method is based on the [Fourier shift theorem](#). Let the two images g_a and g_b be circularly-shifted versions of each other:

$$g_b(x, y) \stackrel{\text{def}}{=} g_a((x - \Delta x) \bmod M, (y - \Delta y) \bmod N)$$

(where the images are $M \times N$ in size).

Then, the discrete Fourier transforms of the images will be shifted relatively in [phase](#):

$$\mathbf{G}_b(u, v) = \mathbf{G}_a(u, v) e^{-2\pi i (\frac{u\Delta x}{M} + \frac{v\Delta y}{N})}$$

One can then calculate the normalized cross-power spectrum to factor out the phase difference:

$$\begin{aligned} R(u, v) &= \frac{\mathbf{G}_a \mathbf{G}_b^*}{|\mathbf{G}_a \mathbf{G}_b^*|} \\ &= \frac{\mathbf{G}_a \mathbf{G}_a^* e^{2\pi i (\frac{u\Delta x}{M} + \frac{v\Delta y}{N})}}{|\mathbf{G}_a \mathbf{G}_a^* e^{2\pi i (\frac{u\Delta x}{M} + \frac{v\Delta y}{N})}|} \\ &= \frac{\mathbf{G}_a \mathbf{G}_a^* e^{2\pi i (\frac{u\Delta x}{M} + \frac{v\Delta y}{N})}}{|\mathbf{G}_a \mathbf{G}_a^*|} \\ &= e^{2\pi i (\frac{u\Delta x}{M} + \frac{v\Delta y}{N})} \end{aligned}$$

since the magnitude of a [complex exponential](#) always is one, and the phase of $\mathbf{G}_a \mathbf{G}_a^*$ always is zero.

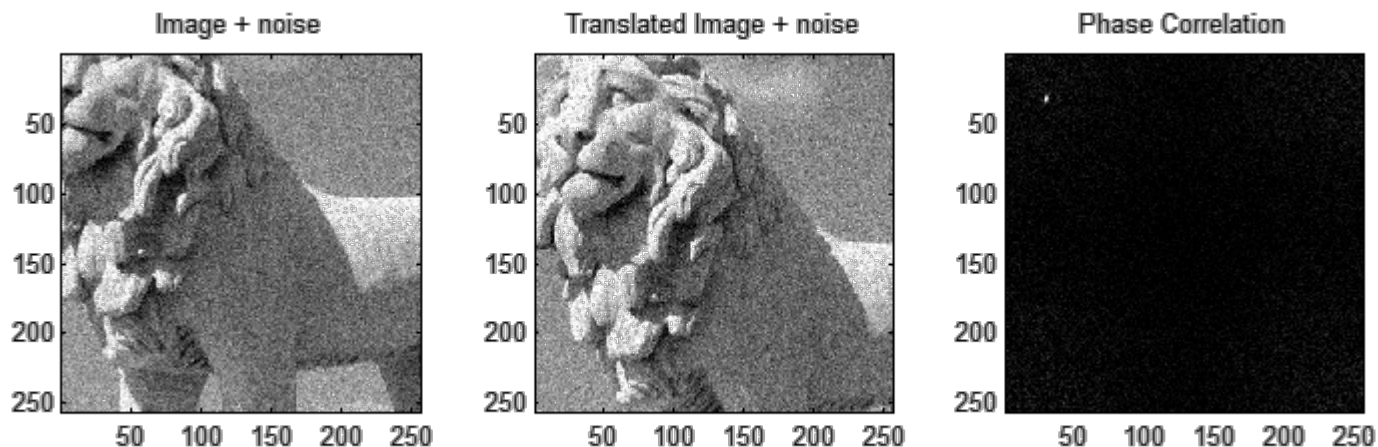
Optical Field: Phase Correlation Method [10]

The inverse Fourier transform of a complex exponential is a Kronecker delta, i.e. a single peak:

$$r(x, y) = \delta(x + \Delta x, y + \Delta y)$$

This result could have been obtained by calculating the cross correlation directly. The advantage of this method is that the discrete Fourier transform and its inverse can be performed using the fast Fourier transform, which is much faster than correlation for large images.

The following image demonstrates the usage of phase correlation to determine relative translative movement between two images corrupted by independent Gaussian noise. The image was translated by (30,33) pixels. Accordingly, one can clearly see a peak in the phase-correlation representation at approximately (30,33).



Optical Field: Lucas–Kanade Method [8]

Assuming that the optical flow (V_x, V_y) is constant in a small window of size $m \times m$ with $m > 1$, which is center at (x, y) and numbering the pixels within as $1 \dots n$, $n = m^2$, a set of equations can be found:

$$\nabla I^T \cdot \mathbf{V} = -I_t \Rightarrow$$

$$\begin{aligned} I_{x1}V_x + I_{y1}V_y &= -I_{t1} \\ I_{x2}V_x + I_{y2}V_y &= -I_{t2} \\ &\vdots \\ I_{xn}V_x + I_{yn}V_y &= -I_{tn} \end{aligned}$$

With this there are more than two equations for the two unknowns and thus the system is over-determined. Hence:

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

or

$$A\vec{v} = -b$$

Optical Flow: Lucas–Kanade Method [8]

To solve the over-determined system of equations, besides other methods, the **least squares** method can also be used:

$$A^T A \vec{v} = A^T (-b) \text{ or } \\ \vec{v} = (A^T A)^{-1} A^T (-b)$$

or

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{x_i} I_{y_i} & \sum I_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix}$$

with the sums running from $i=1$ to n .

The matrix $A^T A$ is often called the **structure tensor** of the image at the point p .

cf.) Harris corner detector



KLT Feature Tracker [11]

The KLT feature tracker is based on two papers:

In the first paper, Lucas and Kanade^[1] developed the idea of a local search using gradients weighted by an approximation to the second derivative of the image.

1. ^ Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

In the second paper Tomasi and Kanade^[2] used the same basic method for finding the registration due to the translation but improved the technique by tracking features that are suitable for the tracking algorithm.

2. ^ Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, April 1991.
4. ^ J. Shi and C. Tomasi (June 1994). "Good Features to Track," . *9th IEEE Conference on Computer Vision and Pattern Recognition*. Springer.
C. Tomasi and T. Kanade (2004). "Detection and Tracking of Point Features" . *Pattern Recognition* 37: 165–168. doi:10.1016/S0031-3203(03)00234-6 .

KLT Feature Tracker [11]

If d is the displacement between two images $F(x)$ and $G(x) = F(x + d)$ then the approximation is made that

$$F'(x) \approx \frac{G(x) - F(x)}{d}$$

so

$$d \approx \frac{G(x) - F(x)}{F'(x)}$$

This approximation to the gradient of the image is only accurate if the displacement of the local area between the two images to be registered is not too large. Lucas and Kanade went on to show an improved derivation of the same formula, considering a local area around a feature

$$d \approx \frac{\sum_x F'(x)[G(x) - F(x)]}{\sum_x F'(x)^2} \quad \leftarrow F'(x) \text{ ㄷ weighting}$$

In implementation the displacement between the feature and its estimated position are iteratively calculated in an iterative **Newton–Raphson**-style search. Smoothing and multiresolution are used to improve the search range.

KLT Feature Tracker [11]

The proposed features would be selected if both the eigenvalues of the gradient matrix were larger than some threshold.

By a very similar derivation, the problem is formulated as

$$\nabla d = e \qquad \longleftarrow \nabla I^T \cdot \mathbf{V} = -I_t$$

where ∇ is the gradient. This is the same as the last formula of Lucas–Kanade above. A local patch is considered a good feature to track if both of the two eigenvalues (λ_1 and λ_2) of ∇ are larger than a threshold.

KLT: An Implementation of the Kanade–Lucas–Tomasi Feature Tracker

<http://www.ces.clemson.edu/~stb/klt/>

Lucas-Kanade Method: A Unifying Framework [12]

Since the Lucas-Kanade algorithm was proposed in 1981 image alignment has become one of the most widely used techniques in computer vision.

Besides optical flow, some of its other applications include

- tracking (Black and Jepson, 1998; Hager and Belhumeur, 1998),
- parametric and layered motion estimation (Bergen et al., 1992),
- mosaic construction (Shum and Szeliski, 2000),
- medical image registration (Christensen and Johnson, 2001),
- face coding (Baker and Matthews, 2001; Cootes et al., 1998).

Lucas-Kanade Method: A Unifying Framework [12]

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template T and the image I warped back onto the coordinate frame of the template:

$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p})) - T(\mathbf{x})]^2. \quad (3)$$

Warping I back to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$ requires interpolating the image I at the sub-pixel locations $\mathbf{W}(\mathbf{x}; \mathbf{p})$. The minimization of the expression in Equation (3) is performed with respect to \mathbf{p} and the sum is performed over all of the pixels \mathbf{x} in the template image $T(\mathbf{x})$.

Let $\mathbf{W}(\mathbf{x}; \mathbf{p})$ denote the parameterized set of allowed warps, where $\mathbf{p} = (p_1, \dots, p_n)^T$ is a vector of parameters.

If we are computing optical flow, for example, the warps $\mathbf{W}(\mathbf{x}; \mathbf{p})$ might be the translations:

$$\mathbf{W}(\mathbf{x}; \mathbf{p}) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix} \quad (1)$$

Lucas-Kanade Method: A Unifying Framework [12]

Minimizing the expression in Equation (1) is a non-linear optimization task even if $\mathbf{W}(\mathbf{x}; \mathbf{p})$ is linear in \mathbf{p} because the pixel values $I(\mathbf{x})$ are, in general, non-linear in \mathbf{x} .

To optimize the expression in Equation (3), the Lucas-Kanade algorithm assumes that a current estimate of \mathbf{p} is known and then iteratively solves for increments to the parameters $\Delta\mathbf{p}$; i.e. the following expression is (approximately) minimized:

$$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x})]^2 \quad (4)$$

with respect to $\Delta\mathbf{p}$, and then the parameters are updated:

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}. \quad (5)$$

These two steps are iterated until the estimates of the parameters \mathbf{p} converge. Typically the test for convergence is whether some norm of the vector $\Delta\mathbf{p}$ is below a threshold ϵ ; i.e. $\|\Delta\mathbf{p}\| \leq \epsilon$.

Lucas-Kanade Method: A Unifying Framework [12]

The Lucas-Kanade algorithm (which is a Gauss-Newton gradient descent non-linear optimization algorithm) is then derived as follows. The non-linear expression in Equation (4) is linearized by performing a first order Taylor expansion on $I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p}))$ to give:

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - T(\mathbf{x}) \right]^2. \quad (6)$$

$\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ is the *gradient* of image I evaluated at $\mathbf{W}(\mathbf{x}; \mathbf{p})$

The term $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ is the *Jacobian* of the warp.

$\mathbf{W}(\mathbf{x}; \mathbf{p}) = (W_x(\mathbf{x}; \mathbf{p}), W_y(\mathbf{x}; \mathbf{p}))^T$ then:

$$\frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{pmatrix}.$$

Lucas–Kanade Method: A Unifying Framework [12]

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - T(\mathbf{x}) \right]^2. \quad (6)$$

Minimizing the expression in Equation (6) is a least squares problem and has a closed form solution which can be derived as follows. The partial derivative of the expression in Equation (6) with respect to $\Delta \mathbf{p}$ is:

$$2 \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[I(\mathbf{W}(\mathbf{x}; \mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - T(\mathbf{x}) \right] \quad (9)$$

we refer to $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ as the *steepest descent* images.

Lucas-Kanade Method: A Unifying Framework [12]

Setting this expression to equal zero and solving gives the closed form solution for the minimum of the expression in Equation (6) as:

$$\Delta \mathbf{p} = H^{-1} \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))] \quad (10)$$

where H is the $n \times n$ (Gauss-Newton approximation to the) *Hessian* matrix:

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]. \quad (11)$$

we refer to $\sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))]$ as the steepest descent parameter updates.
SD

Equation (10) then expresses the fact that the parameter updates $\Delta \mathbf{p}$ are the steepest descent parameter updates multiplied by the inverse of the Hessian matrix.

The Lucas-Kanade algorithm [13] then consists of iteratively applying Equations (10) and (5).

Lucas-Kanade Method: A Unifying Framework [12]

The Lucas-Kanade Algorithm

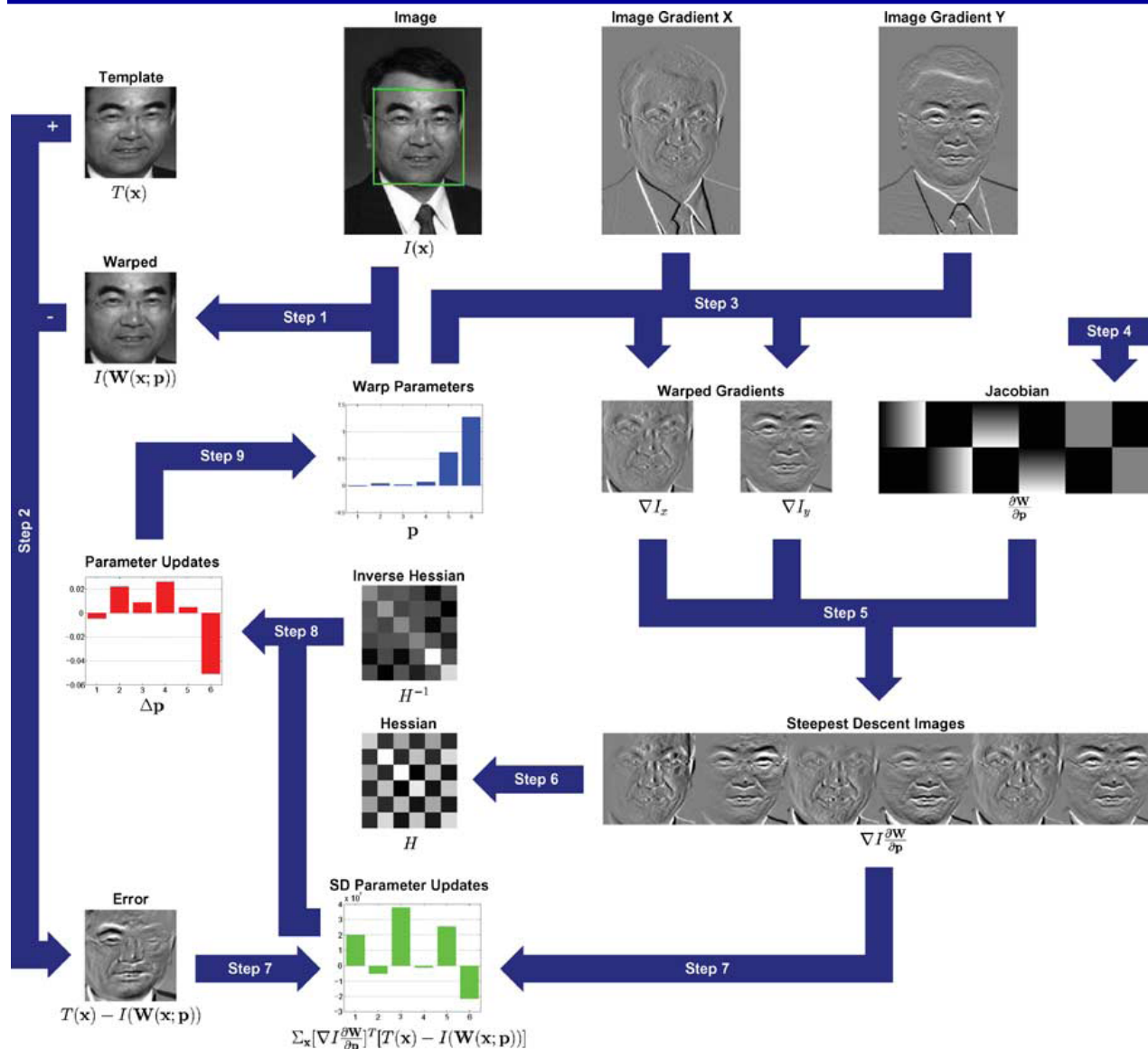
Iterate:

- (1) Warp I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$ to compute $I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
- (2) Compute the error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
- (3) Warp the gradient ∇I with $\mathbf{W}(\mathbf{x}; \mathbf{p})$
- (4) Evaluate the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $(\mathbf{x}; \mathbf{p})$
- (5) Compute the steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$
- (6) Compute the Hessian matrix using Equation (11)
- (7) Compute $\sum_{\mathbf{x}} [\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}]^T [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x}; \mathbf{p}))]$
- (8) Compute $\Delta \mathbf{p}$ using Equation (10)
- (9) Update the parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$

until $\|\Delta \mathbf{p}\| \leq \epsilon$

Figure 1: The Lucas-Kanade algorithm [13] consists of iteratively applying Equations (10) & (5) until the estimates of the parameters \mathbf{p} converge. Typically the test for convergence is whether some norm of the vector $\Delta \mathbf{p}$ is below a user specified threshold ϵ . Because the gradient ∇I must be evaluated at $\mathbf{W}(\mathbf{x}; \mathbf{p})$ and the Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ must be evaluated at \mathbf{p} , all 9 steps must be repeated in every iteration of the algorithm.

Lucas-Kanade Method: A Unifying Framework [12]



- (1) Warp I with $W(x; p)$ to compute $I(W(x; p))$
- (2) Compute the error image $T(x) - I(W(x; p))$
- (3) Warp the gradient ∇I with $W(x; p)$
- (4) Evaluate the Jacobian $\frac{\partial W}{\partial p}$ at $(x; p)$
- (5) Compute the steepest descent images $\nabla I \frac{\partial W}{\partial p}$
- (6) Compute the Hessian matrix using Equation (11)
- (7) Compute $\sum_x [\nabla I \frac{\partial W}{\partial p}]^T [T(x) - I(W(x; p))]$
- (8) Compute Δp using Equation (10)
- (9) Update the parameters $p \leftarrow p + \Delta p$

References

1. Richard Szeliski, “Dense motion estimation,” *Computer Vision: Algorithms and Applications*, 19 June 2009 (draft), pp. 383-426.
2. Emanuele Trucco, Alessandro Verri, “8. Motion,” *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, New Jersey 1998, pp.177-218.
3. Wikipedia, “Motion field,” available on www.wikipedia.org.
4. Wikipedia, “Optical flow,” available on www.wikipedia.org.
5. Alessandro Verri, Emanuele Trucco, “Finding the Epipole from Uncalibrated Optical Flow,” BMVC 1997, available on <http://www.bmva.ac.uk/bmvc/1997/papers/052/bmvc.html>.
6. Wikipedia, “Parallax,” available on www.wikipedia.org.
7. Jae Kyu Suhr, Ho Gi Jung, Kwanghyuk Bae, Jaihie Kim, “Outlier rejection for cameras on intelligent vehicles,” *Pattern Recognition Letters* 29 (2008) 828-840.
8. Wikipedia, “Lucas-Kanade Optical Flow Method,” available on www.wikipedia.org.
9. Wikipedia, “Aperture Problem,” available on www.wikipedia.org.
10. Wikipedia, “Phase correlation,” available on http://en.wikipedia.org/wiki/Phase_correlation
11. Wikipedia, “Kanade-Lucas-Tomasi feature tracker,” available on http://en.wikipedia.org/wiki/Kanade%20%93Lucas%20%93Tomasi_feature_tracker
12. Simon Baker, Iain Matthews, “Lucas-Kanade 20 Years: A Unifying Framework,” *International Journal of Computer Vision*, 56(3), 221-255, 2004.