Motion Analysis

We are interested in the visual information that can be extracted from the content changes occurring in an image sequence. An image sequence consists of a series of images (frames) acquired at consecutive discrete time instants. They are acquired at the same or different view points.

Motion is important since it represents spatial changes over time, which is crucial for understanding the dynamic world.

Motion Analysis (cont'd)

- **Image motion** characterizes image content change at different frames because of the relative 3D motion between the camera and the scene.
- We are either dealing with a static scene yet a moving camera or a stationary camera but dynamic scene or both.
- The goal of motion analysis is to compute the image motion and use it as a visual cue to perform 3D dynamics analysis tasks, including 3D motion estimation, 3D structure reconstruction, object detection, scene segmentation, etc...

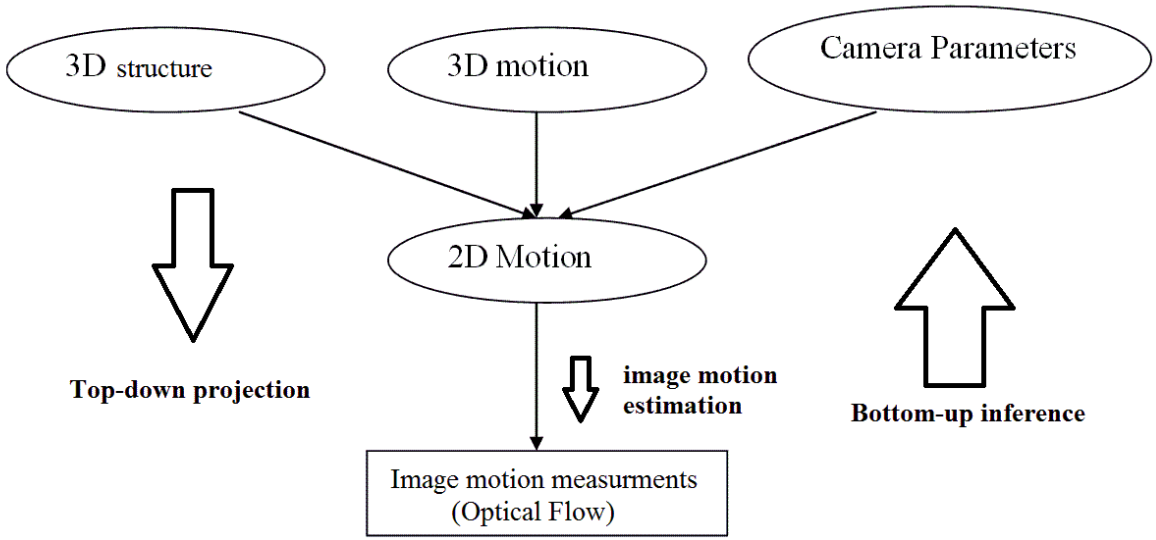
Motion Analysis (cont'd)

Motion analysis can answer the following questions

- How many moving objects are there ?
- Which directions are they moving ?
- How fast are they moving
- What are the structures of the moving objects?

A Graphical Model for Motion Modeling and Analysis

A graphical model shows the causal relationships between 3D structure, 3D motion, camera parameters, and the image motion.



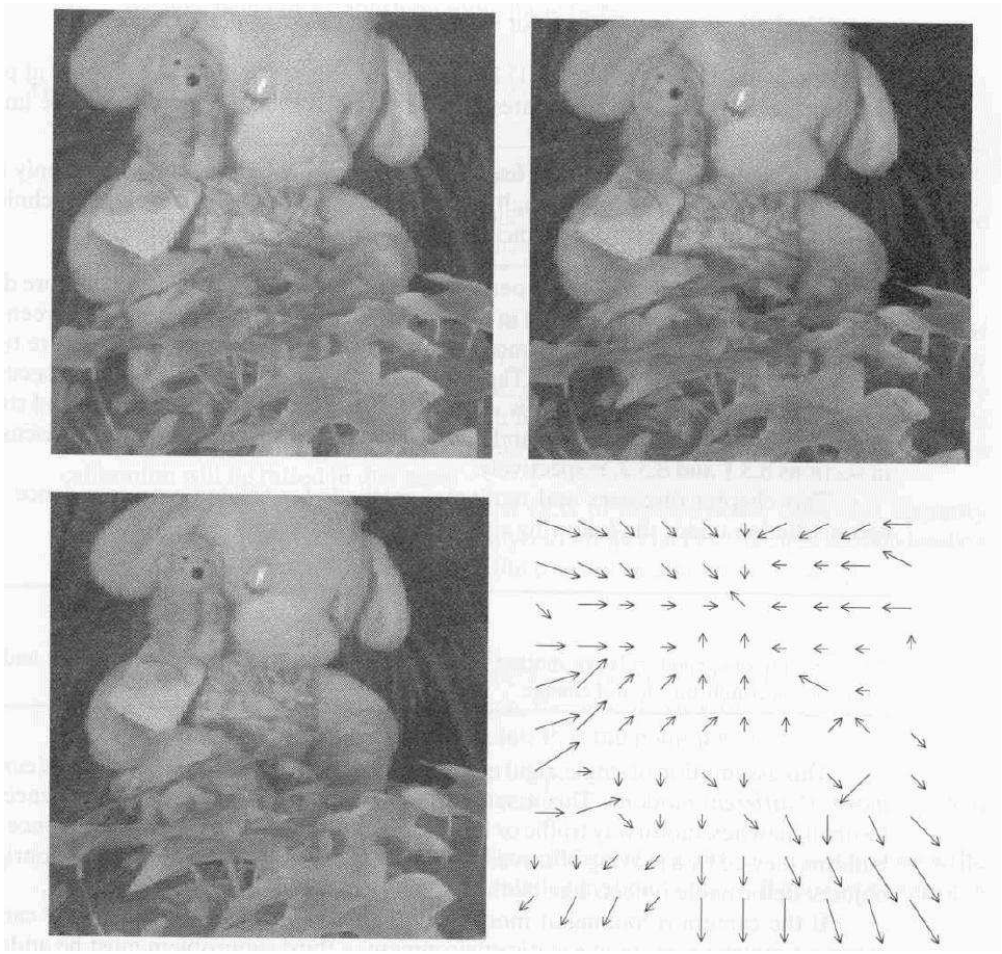
- Top-down projection: produce image motion from 3D structure, motion, and camera parameters via a projection model

- Image motion estimation
- Bottom-up inference: estimate the 3D motion and structure from the estimated image motion measurements.

Tasks in Motion Analysis

- Image motion estimation
- 3D motion and 3D structure estimation from the estimated image motion
- motion based segmentation-divide a scene into different regions (motion analysis), each of which may be characterized by different motion characteristics.

See fig. 8.3, where the foreground and the background move in different directions. The foreground moves toward the camera while the background (the teddy bear) moves away from the camera.



Motion Analysis v.s. Stereo

The task of 3D motion and 3D structure estimation from an image sequence is similar to the stereo problem. They are different however:

- much smaller disparities for motion between consecutive frames due to small spatial differences between consecutive frames.
- the relative 3D movement between camera and the scene may be caused by multiple 3D rigid transformations since the scene cannot be modeled as a single rigid entity. It may contain multiple rigid objects with different motion characteristics.

Motion Analysis v.s. Stereo (cont'd)

- Motion analysis can take advantage of small temporal and spatial changes between two consecutive frames. Specifically, the past history of the target motion and appearance (intensity distribution pattern) may be used to predict current motion. This is referred to as *tracking*.
- The reconstruction is often not accurate due to small baseline distance between two consecutive frames. Like the stereo, the main bottleneck for motion analysis is to determine correspondences.

Image Motion Analysis

There are two methods: differential methods based on time derivatives (image flow) and matching methods based on tracking. The former leads to dense correspondences at each pixel while the latter leads to sparse correspondences.

Basics in Motion Analysis

For subsequent discussion, we assume there is only one rigid relative motion between camera and the object.

Image motion field is defined as the 2D vector field of velocities of the image points, induced by relative 3D motion between the view camera and the observed scene. It can also be interpreted as the projection of 3D velocity field on the image plane.

Let $P = (X, Y, Z)^T$ be a 3D point relative to the *camera frame* and $p = (c, r)^T$ be the projection of P in the image frame. Hence,

$$\lambda \begin{pmatrix} p \\ 1 \end{pmatrix} = WP$$

$$\begin{aligned} c &= \frac{W_1}{Z} P = \frac{f s_x X}{Z} + c_0 \\ r &= \frac{W_1}{Z} P = \frac{f s_y Y}{Z} + r_0 \end{aligned}$$

where W_1 and W_2 be the first and second rows of W. They can be alternatively written as

$$
\begin{aligned}
c - c_0 &= \frac{f s_x X}{Z} \\
r - r_0 &= \frac{f s_y Y}{Z}
\end{aligned}
$$

The 3D motion can be computed as

$$V = \frac{dP}{dt}$$

Assume V is composed of translational movement

$T = (t_X, t_Y, t_Z)$ and a rotational movement $\omega = (\omega_x, \omega_y, \omega_z)$, then

$$V = \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = -T - \omega \times P \quad (1)$$

$$= \begin{pmatrix} -t_X + \omega_z Y - \omega_y Z \\ -t_Y + \omega_x Z - \omega_z X \\ -t_Z + \omega_y X - \omega_x Y \end{pmatrix} \quad (2)$$

where $V_x = \frac{dX}{dt}$, $V_y = \frac{dY}{dt}$, and $V_z = \frac{dZ}{dt}$. The motion field in the image is resulted from projecting V onto the image plane.

The motion of image point p is characterized as $v = (v_c, v_r)^T$:

$$
\begin{aligned}
v_c &= \frac{dc}{dt} = \frac{d\frac{W_1 P}{Z}}{dt} = W_1[\frac{\frac{dP}{dt}}{Z} - \frac{P\frac{dZ}{dt}}{Z^2}] = W_1[\frac{V}{Z} - \frac{PV_z}{Z^2}] \\
&= \frac{W_1 V}{Z} - \frac{cV_z}{Z} \\
&= \frac{(-t_X + \omega_z Y - \omega_y Z) f s_x + c_0(-t_Z + \omega_y X - \omega_x Y)}{Z} \\
&- \frac{c(-t_Z + \omega_y X - \omega_x Y)}{Z} \\
&= \frac{-t_X f s_x - c_0 t_Z + c t_Z}{Z} \\
&+ \frac{\omega_z Y f s_x - \omega_y Z f s_x + c_0 \omega_y X - c_0 \omega_x Y - c\omega_y X + c\omega_x Y}{Z} \\
&= \frac{-t_X f s_x + (c - c_0) t_Z}{Z} - \omega_y f s_x \\
&+ \frac{\omega_z Y f s_x + (c - c_0)(\omega_x Y - \omega_y X)}{Z}
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
&= \frac{-t_X f s_x + (c - c_0) t_Z}{Z} - \omega_y f s_x \\
&+ (r - r_0)\omega_z \frac{s_x}{s_y} + \frac{(c - c_0)(r - r_0)\omega_x}{f s_y} - \frac{(c - c_0)^2 \omega_y}{f s_x} \\
&= \underbrace{(c - c_0 - \frac{f s_x t_X}{t_Z})\frac{t_Z}{Z}}_{\text{translational motion}} \\
&- \underbrace{\omega_y f s_x + (r - r_0)\omega_z \frac{s_x}{s_y} + \frac{(c - c_0)(r - r_0)\omega_x}{f s_y} - \frac{(c - c_0)^2 \omega_y}{f s_x}}_{\text{rotational motion}}
\end{aligned}
\tag{4}
$$

Similarly, we have

$$\begin{aligned} v_r &= \frac{dr}{dt} = \frac{-t_Y f s_y + (r - r_0) t_Z}{Z} + \omega_x f s_y \\ &- (c - c_0)\omega_z \frac{s_y}{s_x} - \frac{(c - c_0)(r - r_0)\omega_y}{f s_x} + \frac{(r - r_0)^2 \omega_x}{f s_y} \\ &= (r - r_0 - \frac{f s_y t_Y}{t_Z})\frac{t_Z}{Z} + \omega_x f s_y \\ &- (c - c_0)\omega_z \frac{s_y}{s_x} - \frac{(c - c_0)(r - r_0)\omega_y}{f s_x} + \frac{(r - r_0)^2 \omega_x}{f s_y} \end{aligned} \quad (5)$$

Note the motion field is the sum of two components, one for translation and the other for rotation only, i.e.,

$$v_c^t = \frac{-t_X f s_x + (c - c_0) t_Z}{Z} \quad (6)$$

$$v_r^t = \frac{-t_Y f s_y + (r - r_0) t_Z}{Z} \quad (7)$$

the translational motion component is inversely proportional to the depth Z.

The angular motion component does not carry information of object depth (no Z in the equations). Structures are recovered based on translational motion.

$$v_c^\omega = -\omega_y f s_x + (r - r_0)\omega_z \frac{s_x}{s_y} + \frac{(c - c_0)(r - r_0)\omega_x}{f s_y} - \frac{(c - c_0)^2 \omega_y}{f s_x}$$

$$v_r^\omega = \omega_x f s_y - (c - c_0)\omega_z \frac{s_y}{s_x} - \frac{(c - c_0)(r - r_0)\omega_y}{f s_x} + \frac{(r - r_0)^2 \omega_x}{f s_y}$$

Alternative Derivations of the Motion Projection Equation

From the projection equation,

$$\lambda \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = W \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{8}$$

We have

$$\begin{pmatrix} c \\ r \end{pmatrix} = \frac{1}{\lambda} W_2 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{1}{Z} W_2 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{9}$$

where W_2 are the first two rows of W.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \lambda W^{-1} \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = Z W^{-1} \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} \tag{10}$$

$$
\begin{aligned}
\frac{dp}{dt} &= \frac{d\begin{pmatrix} c \\ r \end{pmatrix}}{dt} = \frac{\overbrace{d(\frac{W_2}{Z}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix})}^{\text{Eq.9}}}{dt} \\
&= \frac{W_2}{Z}\frac{d\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}}{dt} - \frac{W_2\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}}{Z^2}\frac{dZ}{dt} \\
&= \frac{W_2}{Z}V - \begin{pmatrix} c \\ r \end{pmatrix}\frac{V_z}{Z} \qquad (11)
\end{aligned}
$$

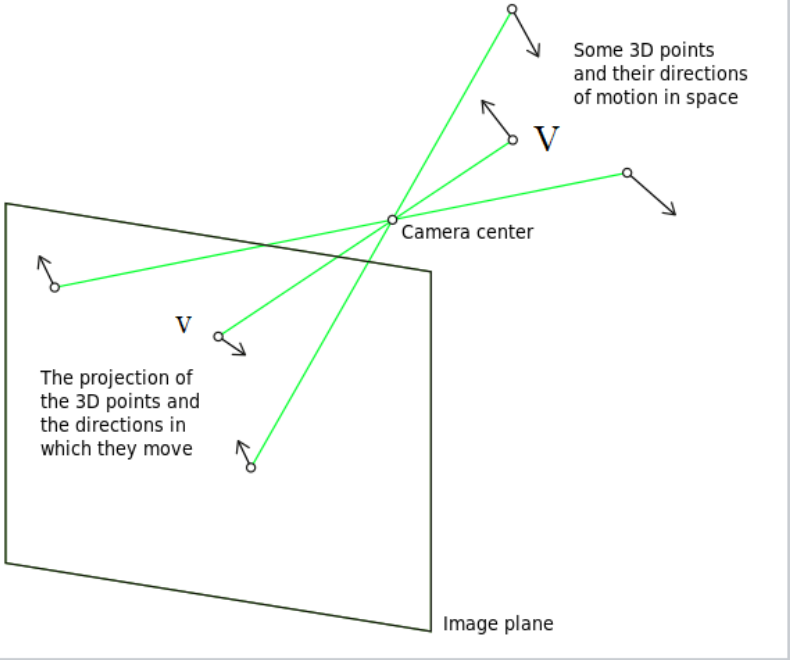
Eq. 11 is the motion projection equation. Introducing the rotational and

translational motion, Eq. 11 can be written as

$$\frac{dp}{dt} = \frac{W_2}{Z}(-T - \omega \times \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}) - \begin{pmatrix} c \\ r \end{pmatrix} \frac{-t_z + \omega_y X - \omega_x Y}{Z}$$

$$
\begin{aligned}
&= \frac{-W_2T + \begin{pmatrix} c \\ r \end{pmatrix} t_z}{Z} + \frac{-W_2\omega \times \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - \begin{pmatrix} c \\ r \end{pmatrix} \begin{pmatrix} \omega_y \\ -\omega_x \end{pmatrix}^T \begin{pmatrix} X \\ Y \end{pmatrix}}{Z} \\
&= \underbrace{\frac{-W_2T + \begin{pmatrix} c \\ r \end{pmatrix} t_z}{Z}} \\
&\quad \text{translational motion that contains Z} \\
&- \underbrace{W_2\omega \times W^{-1} \overbrace{\begin{pmatrix} c \\ r \\ 1 \end{pmatrix}}^{Eq.10} - \begin{pmatrix} c \\ r \end{pmatrix} \begin{pmatrix} \omega_y \\ -\omega_x \\ 0 \end{pmatrix}^T W^{-1} \overbrace{\begin{pmatrix} c \\ r \\ 1 \end{pmatrix}}^{Eq.10}}_{\text{rotational motion that contains no Z}}
\end{aligned} \tag{12}
$$

Motion Field Illustration (from wikipedia)



An illustration of some 3D points and their corresponding image points as described by the pinhole camera model. As the 3D points are moving in space, the corresponding image points are also moving. The motion field consists of the motion vectors in the image for all points in the image.

Motion Field: pure translation

Under pure relative translation movement between the camera and the scene, i.e., $\omega = 0$, we have

$$\begin{aligned} v_c^t &= \frac{-t_X f s_x + (c - c_0) t_Z}{Z} \\ v_r^t &= \frac{-t_Y f s_y + (r - r_0) t_Z}{Z} \end{aligned} \tag{13}$$

Assume $t_Z \neq 0$ and $p_e = (c_e, r_e)^T$ such that

$$c_e = \frac{f s_x t_X}{t_Z} + c_0 \tag{14}$$

$$r_e = \frac{f s_y t_Y}{t_Z} + r_0 \tag{15}$$

Equation 13 can be rewritten as

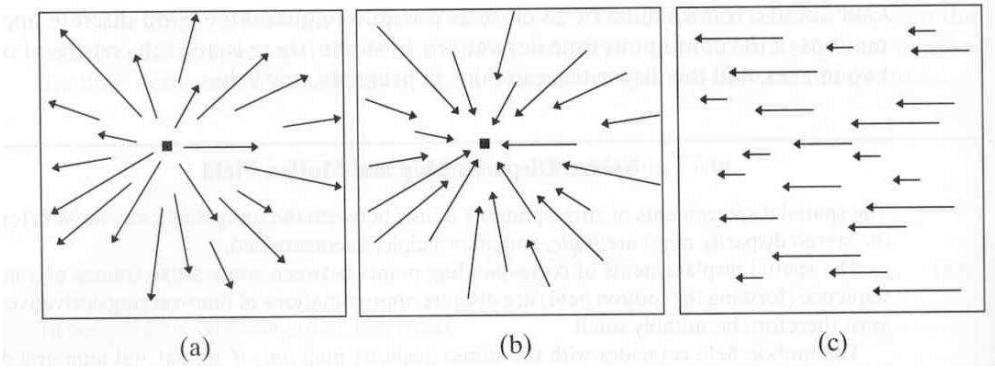
$$
\begin{aligned}
v_c^t &= (c - c_e)\frac{t_Z}{Z} \\
v_r^t &= (r - r_e)\frac{t_Z}{Z}
\end{aligned} \tag{16}
$$

From equation 16, we can conclude

- the motion field vectors for all pixels under pure 3D translation is radial, all going through point p_e as p_e is independent of each pixel.
- the motion field magnitude is inversely proportional to depth but proportional to the distance to p_e
- the motion field radiates from a common origin p_e, and if $t_z > 0$ (move away from the camera), it radiates towards p_e since as object moves away from the camera, its size shrinks and the image point moves towards the p_e. On the other

hand, as the object moves towards the camera, i.e., $t_z < 0$, the object size increases and each image point, therefore, moves away from p_e. The motion field radiates outward from p_e.

- when $t_z = 0$, i.e., movement is limited to x and y directions, then motion field is
  parallel to each other other since $v_c = -\frac{f s_x t_X}{Z}$ and $v_r = -\frac{f s_y t_Y}{Z}$



The above conclusions allow us to infer 3D motion from their 2D motion field, given some knowledge about the 3D motion (such as translational motion).

Motion Field: Planar Motion

The relative planar motion between the camera and scene induces a motion field of quadratic polynomial of the image coordinates. The same motion field can be produced by two different planar surfaces undergoing different 3D motions due to special symmetry with the polynomial coefficients. Therefore, 3D motion and structure recovery can not be based on planar motion (same as camera calibration).

Motion Field Estimation

Given a sequence of images, estimate $v = (v_c, v_r)$ for every pixel at (c,r).

The image brightness constancy equation. The intensity of each pixel at a point (c,r) is a function of c, r, and t, and the coordinates (c,r) also vary with time t. We can hence write the intensity at pixel (c,r) as $I(c(t), r(t), t)$.

The image brightness constancy assumes that $I(c(t), r(t), t)$ at two consecutive times remains the same, i.e.,
$I(c(t), r(t), t) = I(c(t+1), r(t+1), t+1)$, which means the intensity change between t and t+1 is 0, i.e.,

$$\frac{dI}{dt} = 0$$

this constraint is completely satisfied only under 1) the motion is

translational motion; or 2) illumination direction is parallel to the angular velocity for lambertian surface. This can be verified by assuming the lambertian surface model

$$I = \rho n^T L$$

Hence,

$$\begin{aligned} \frac{dI}{dt} &= \rho(\frac{dn}{dt})^T L \\ &= \rho(\omega \times n)^T L \end{aligned}$$

Hence, $\frac{dI}{dt} = 0$ only when either $\omega = 0$ or ω and n are parallel to each other.

Via rule of differentiation, we have

$$\frac{dI}{dt} = \frac{\partial I}{\partial c}\frac{dc}{dt} + \frac{\partial I}{\partial r}\frac{dr}{dt} + \frac{\partial I}{\partial t} = 0 \tag{17}$$

$\frac{\partial I}{\partial c}$ and $\frac{\partial I}{\partial r}$ represent spatial image gradient while $\frac{\partial I}{\partial t}$ represents temporal image gradient. Let $I_c = \frac{\partial I}{\partial c}$, $I_r = \frac{\partial I}{\partial r}$, and $I_t = \frac{\partial I}{\partial t}$, we have the image brightness constancy equation as

$$I_c v_c + I_r v_r + I_t = 0 \tag{18}$$

it can be alternatively written as

$$(\bigtriangledown I)^T v + I_t = 0 \tag{19}$$

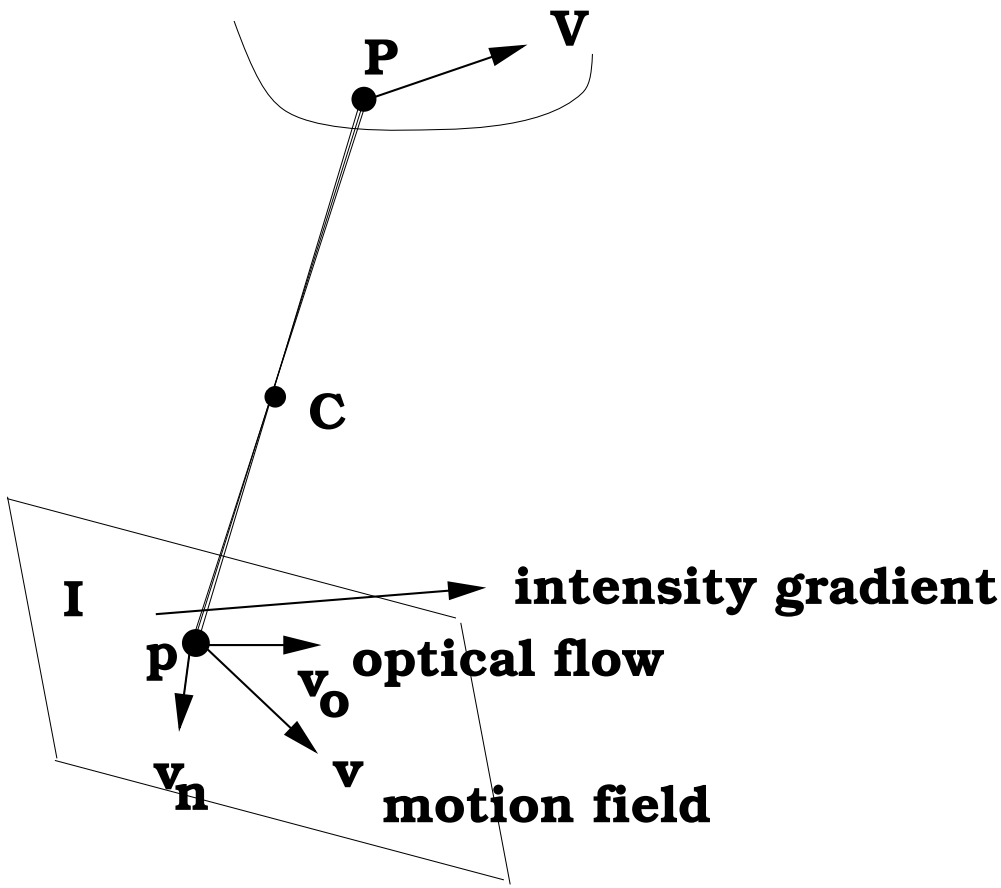
$\bigtriangledown I = (I_c, I_r)^T$ is image intensity gradient. This equation may be used to estimate the motion field $v = (v_c, v_r)$. Note this equation

has no constraint on v when it is orthogonal to $\bigtriangledown I$. So equation 18 can only determine motion flow component in the direction of the intensity gradient, i.e., the projection of motion field v in the gradient direction (due to the dot product). This special motion field is called *optical flow*. So, optical flow is always parallel to image gradient.

Optical Flow

Optical flow is a vector field in the image that represents a projection of the image motion field, along the direction of image gradient.

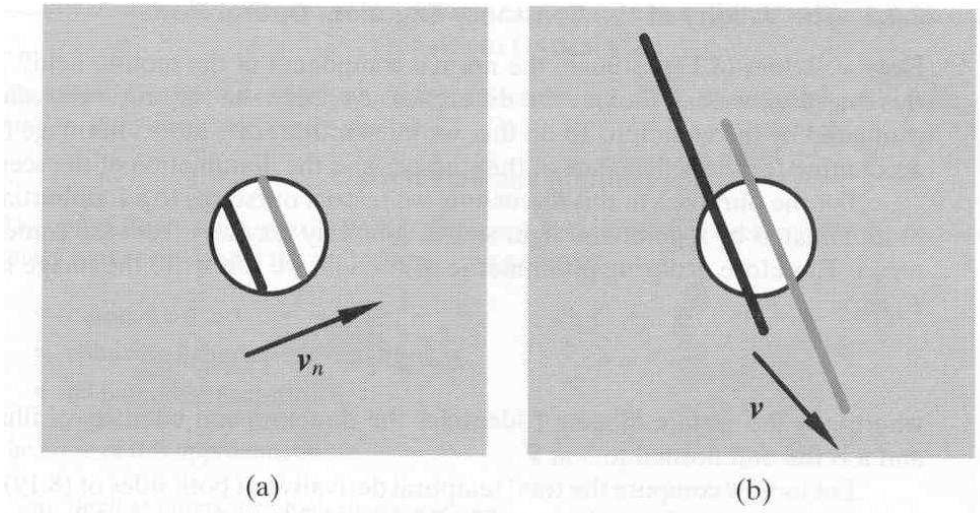
Follow the same derivation, derive the optical flow equation for 1D signal $I(x,t)$ using $dI(x,t)/dt = 0$, Can it be uniquely solved?



motion field v.s. optical flow

Aperture Problem

The component of motion field in the direction orthogonal to the spatial image gradient is not constrained by the image brightness constancy equation. Different 3D motions may produce the same 2D optical flows but different motion fields in the direction orthogonal to the image gradient.



Time to Contact Estimation

The time to contact is usually expressed in terms of the speed and the distance of the considered obstacle, i.e.,

$$\tau = -\frac{Z}{\frac{dZ}{dt}}$$

where Z is the distance to the object and dZ/dt is the speed to the object. Optical flow can be used to estimate dZ/dt. However, with a monocular camera the distance Z is generally unknown. Its possible to derive (1) by using a characteristic size of the obstacle in the image and by assuming that the obstacle is planar and parallel to the image plane (i.e. affine image conditions)

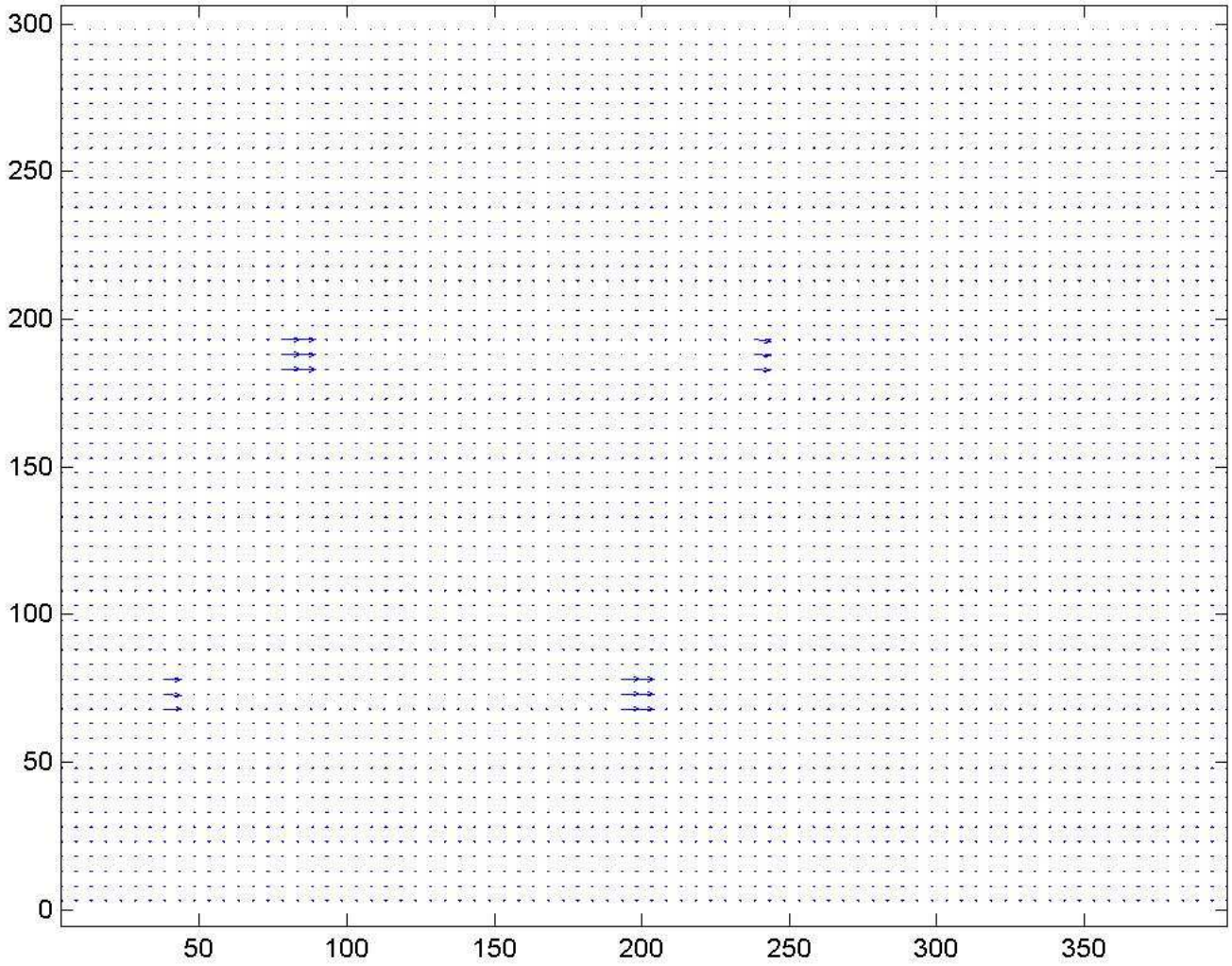
$$\tau = -\frac{\rho}{\frac{d\rho}{dt}}$$

where ρ is the size (or the scale) of the object in the image and $d\rho/dt$ the time derivative of this scale. This equation is more appropriate as the size can be obtained directly in the image space. This reformulates the problem as a problem of estimating the size obstacle size, as well as the rate of change of size. Note that the TTC does not rely on the absolute size of the object in the image sequence, but on the relative change in scale from one frame to another. As a consequence, the TTC computation is not dependent on camera optics or the object size, only is dependent on the depth distance and the camera velocity. More from the paper A Comparison of Three Methods for Measure of Time to Contact at

http://digital.csic.es/bitstream/10261/30110/1/comparisonThe paper is also available in the motion folder.

Aperture Problem Example

In this example, the motion is horizontal while the gradients are vertical except for the vertical edges at ends. Optical flows are not detected except for the vertical edges. This may be used to do motion-based edge detection.


300
250
200
150
100
50
0
50
100
150
200
250
300
350

OF Estimation

Given the image constancy equation, OF estimation can be carried our using a local approach or a global approach. Local approach estimates the optical flow for each pixel individually and it can be further divided into first order and second order methods. The global approach estimates the optical flows for all pixels simultaneously. We first introduce two local methods. We can then discuss Horn and Schuck method as a global approach.

Local OF Estimation: Lucas-Kanade Method

To estimate optical flow, we need an additional constraint since equation 18 only provides one equation for 2 unknowns.

For each image point $p = (c, r)$ and a $K \times K$ neighborhood $R(c, r)$, where (c, r) is the center of R, assume every point in the neighborhood, including (c,r), has the same optical flow $v = (v_c, v_r)$ (note this is a smoothness constraint and may not hold near the edges of the moving objects).

$$I_c(c_i, r_i)v_c + I_r(c_i, r_i)v_r + I_t(c_i, r_i) = 0 \qquad (c_i, r_i) \in R(c, r)$$

(v_c, v_r) can be estimated via minimizing

$$\begin{aligned}
\epsilon^2 &= \sum_{(c_i,r_i)\in R(c,r)} [I_c(c_i,r_i)v_c + I_r(c_i,r_i)v_r + I_t(c_i,r_i)]^2 \\
&= \sum_{(c_i,r_i)\in R(c,r)} [(I_c(c_i,r_i), I_r(c_i,r_i)) \begin{pmatrix} v_c \\ v_r \end{pmatrix} + I_t(c_i,r_i)]^2 \\
&= ||Av + b||_2^2 \qquad (20)
\end{aligned}$$

where

$$\begin{aligned}
A &= \begin{bmatrix} I_c(c_1,r_1) & I_r(c_1,r_1) \\ I_c(c_2,r_2) & I_r(c_2,r_2) \\ \vdots & \vdots \\ I_c(c_N,r_N) & I_r(c_N,r_N) \end{bmatrix} \\
b &= [I_t(c_1,r_1), I_t(c_2,r_2), \ldots, I_t(c_N,r_N)]^T
\end{aligned}$$

The least-squares solution to $v(x, y)$ is

$$v(x, y) = -(A^T A)^{-1} A^T b \tag{21}$$

see the CONSTANT_FLOW algorithm in Trucoo's book. Note this technique often called the Lucas-Kanade method. Its advantages include simplicity in implementation and only need first order image derivatives. The spatial and temporal image derivatives can be computed using a gradient operator (e.g., Sobel) and are often preceded with a Gaussian smoothing operation. Note the algorithm only applies to rigid motion. For non-rigid motion, there is a paper [3], that extends the method to non-rigid motion estimation.

Note the algorithm can be improved by incorporating a weight with each point in the region R such that points closer to the

center receive more weight than points far away from the center.

Image Brightness Change Constancy

Besides assuming brightness constancy while objects are in motion, we can assume both spatial and temporal image brightness change to be constant, i.e., image brightness change rates in c, r, and t are constant. This represents a relaxation of the image brightness constancy assumption.

Mathematically, these constraints can be formulated as follows: $\frac{dI}{dt} = C$, $\frac{d^2I}{dtdc} = 0$, $\frac{d^2I}{dtdr} = 0$, $\frac{d^2I}{dtdt} = 0$. Applying them to equation 17 yields the four optical flow equations are

$$
\begin{aligned}
v_c I_c + v_r I_r + I_t &= C \\
v_c I_{cc} + v_r I_{rc} + I_{tc} &= 0 \\
v_c I_{cr} + v_r I_{rr} + I_{tr} &= 0 \\
v_c I_{ct} + v_r I_{rt} + I_{tt} &= 0
\end{aligned} \tag{22}
$$

Assuming C is known (or a hyper-parameter to tune), this yields four equations (in fact, we can use the last three equations) for two unknowns $v = (v_c, v_r)$. They can therefore be solved using a linear least squares method by minimizing

$$\|Av - b\|_2^2 \tag{23}$$

where

$$A = \begin{pmatrix} I_c & I_r \\ I_{cc} & I_{cr} \\ I_{rc} & I_{rr} \\ I_{tc} & I_{tr} \end{pmatrix} \quad b = - \begin{pmatrix} I_t - C \\ I_{ct} \\ I_{rt} \\ I_{tt} \end{pmatrix}.$$

$$v = (A^T A)^{-1} A^T b \tag{24}$$

Image Brightness Change Constancy (cont'd)

If C is unknown, we can solve for the vector $v = (v_c, v_r, C)$ via a system of linear equations

$$A = \begin{pmatrix} I_c & I_r & -1 \\ I_{cc} & I_{cr} & 0 \\ I_{rc} & I_{rr} & 0 \\ I_{tc} & I_{tr} & 0 \end{pmatrix} \quad b = - \begin{pmatrix} I_t \\ I_{ct} \\ I_{rt} \\ I_{tt} \end{pmatrix}.$$

$$v = (A^T A)^{-1} A^T b \tag{25}$$

As a project, compare this approach with the original approach where $C = 0$ and see if it improves over the original approach.

First order method v.s. second order method

Comparing the two methods, we have the following observations:

- the first order method is built on the image brightness constancy assumption and further assumes the optical flow vectors are the same in a local neighborhood. The second order method assumes image brightness change constancy. It does not need assume the optical flow vectors are the same in a local neighborhood.
- The first order method involves first order image derivatives, while the second order method requires both first and second order image derivatives. Second order image derivatives are sensitive to image noise.

Computing Image Derivatives

Traditional approach to compute intensity derivatives involves numerical approximation of continuous differentiations.

First order image derivatives:

$$\begin{aligned} I_c(c,r,t) &= I(c,r,t) - I(c-1,r,t) \\ I_r(c,r,t) &= I(c,r,t) - I(c,r-1,t) \\ I_t(c,r,t) &= I(c,r,t) - I(c,r,t-1) \end{aligned}$$

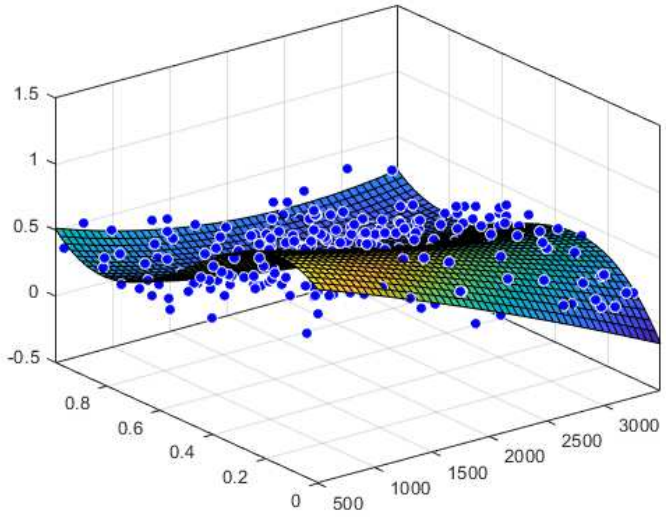
Second order image derivatives:

$$I_{cc}(c,r,t) = \frac{I(c+1,r,t) - 2I(c,r,t) + I(c-1,r,t)}{4}$$

See appendix A.2 of Trucco's book for details.

Image derivatives via Surface Fitting

Use a surface fit with a cubic facet model to the image intensity in a small neighborhood, producing an analytical and continuous image intensity function that approximates image surface at time (c,r,t). Image derivatives can be analytically computed from the coefficients of the cubic facet model.



This yields more robust and accurate image derivatives estimation due to noise suppression via smoothing by function approximation.

Surface Fitting with Cubic Facet Model

Assume the gray level pattern of each small block of $K \times K \times K$ in an image sequence is ideally a canonical 3D cubic polynomial of x, y, t:

$$\begin{aligned} I(c,r,t) &= a_1 + a_2 c + a_3 r + a_4 t + a_5 c^2 + a_6 cr \\ &+ a_7 r^2 + a_8 rt + a_9 t^2 + a_{10} ct + a_{11} c^3 + a_{12} c^2 r \\ &+ a_{13} cr^2 + a_{14} r^3 + a_{15} r^2 t + a_{16} rt^2 + a_{17} t^3 \\ &+ a_{18} c^2 t + a_{19} ct^2 + a_{20} crt, \quad c, r, t \in R, \end{aligned} \tag{26}$$

Each point provides one linear equation w.r.t the cubic coefficients $\mathbf{a} = (a_1, a_2, \ldots, a_{20})^T$. Given a neighborhood of $K \times K \times K$, we can setup up a system of linear equations for the coefficients and solve the coefficients by minimizing $\|D\mathbf{a} - J\|^2$

and is expressed by

$$\mathbf{a} = (D^T D)^{-1} D^T J \tag{27}$$

where

$$D = \begin{pmatrix} 1 & c_1 & r_1 & t_1 & \dots & c_1 r_1 t_1 \\ 1 & c_2 & r_1 & t_1 & \dots & c_2 r_1 t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & c_K & r_1 & t_1 & \dots & c_K r_1 t_1 \\ 1 & c_1 & r_2 & t_1 & \dots & c_1 r_2 t_1 \\ 1 & c_2 & r_2 & t_1 & \dots & c_2 r_2 t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & c_K & r_2 & t_1 & \dots & c_K r_2 t_1 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & c_1 & r_K & t_K & \dots & c_1 r_K t_K \\ 1 & c_2 & r_K & t_K & \dots & c_2 r_K t_K \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & c_K & r_K & t_K & \dots & c_K r_K t_K \end{pmatrix} \qquad J = \begin{pmatrix} I(c_1, r_1, t_1) \\ I(c_2, r_1, t_1) \\ \vdots \\ I(c_K, r_K, t_K) \end{pmatrix}$$

$I(c_i, r_j, t_k)$ is the intensity value at $(c_i, r_j, t_k) \in R^{3\times3}$ and R is the neighborhood volume.

Note while performing surface fitting, the volume should be centered at the current pixel (voxel) being considered and use a local coordinate system, with the center as its origin (0,0). So, for a 3x3x3, neighborhood, the coordinates for x,y and t are respectively: -1 0 1, -1 0 1, and -1 0 1. Do not use the actual coordinates . Hence, D is the same for every pixel, while J varies from pixel to pixel.

Cubic Facet Model (cont'd)

Image derivatives at (0,0,0) are readily available from the cubic facet model. Substituting a_i's into Eq. (23) yields the OFCE's we actually use:

$$A = \begin{pmatrix} a_2 & a_3 \\ 2a_5 & a_6 \\ a_6 & 2a_7 \\ a_{10} & a_8 \end{pmatrix} \quad b = - \begin{pmatrix} a_4 - C \\ a_{10} \\ a_8 \\ 2a_9 \end{pmatrix} \tag{28}$$

Note when estimating the first order image derivatives through cubic fitting for the first order method (LK method), since first order image derivatives are computed not just for the central pixel but also for its neighboring pixels. This means cubic surface fitting needs be performed not only for the center pixel but also

for the pixels in the neighborhood using the same local coordinate system. Based on the surface fitting, we can easily show that

$$I_c(c, r, t) = a_2, I_r(c, r, t) = a_3, I_t(c, r, t) = a_4 \tag{29}$$

They can then be substituted to the A matrix in Eq. 21.

Optical Flow Estimation Algorithm

The input is a sequence of N frames (N=5 typical). Let Q be a square region of $L \times L$ (typically L=5). The steps for estimating optical flow using facet method can be summarized as follows.

- Select an image as central frame (normally the 3rd frame if 5 frames are used)
- For each pixel (excluding the boundary pixels) in the central frame
  - Perform a cubic facet model fit using equation 26 and obtain the 20 coefficients using equation 27.
  - Derive image derivatives using the coefficients and the A matrix and b vector using equation 28 for the second order method or using Eq. 29 for the first order method.
  - Compute image flow using equation 25 or Eq. 21.

- Mark each point with an arrow indicate its flow if its flow magnitude is larger than a threshold.

Set the optical flow vectors to zero for locations where matrix $A^T A$ is singular (or small determinant).

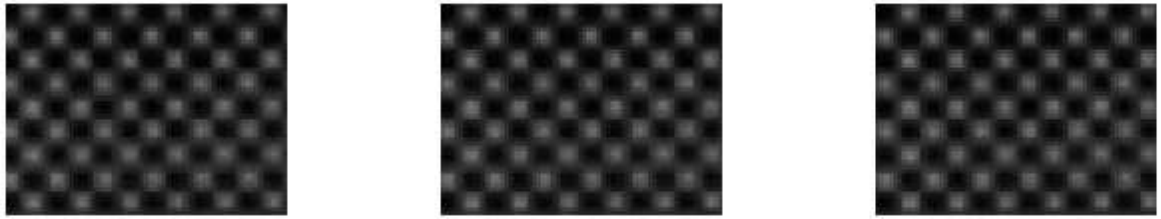
The global approach: Honn and Schuck Method

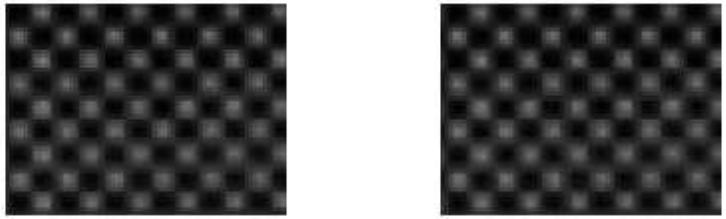
$$
\begin{aligned}
v_c^*(c,r), v_r^*(c,r) &= \arg \min_{v_c(c,r), v_r(c,r)} \sum_c \sum_r \\
&\quad [I_c(c,r)v_c(c,r) + I_r(c,r)v_r(c,r) + I_t(c,r)]^2 \\
&+ \quad \lambda[v_{cc}(c,r) + v_{cr}(c,r) + v_{rr}(c,r) + v_{rc}(c,r)]
\end{aligned}
$$

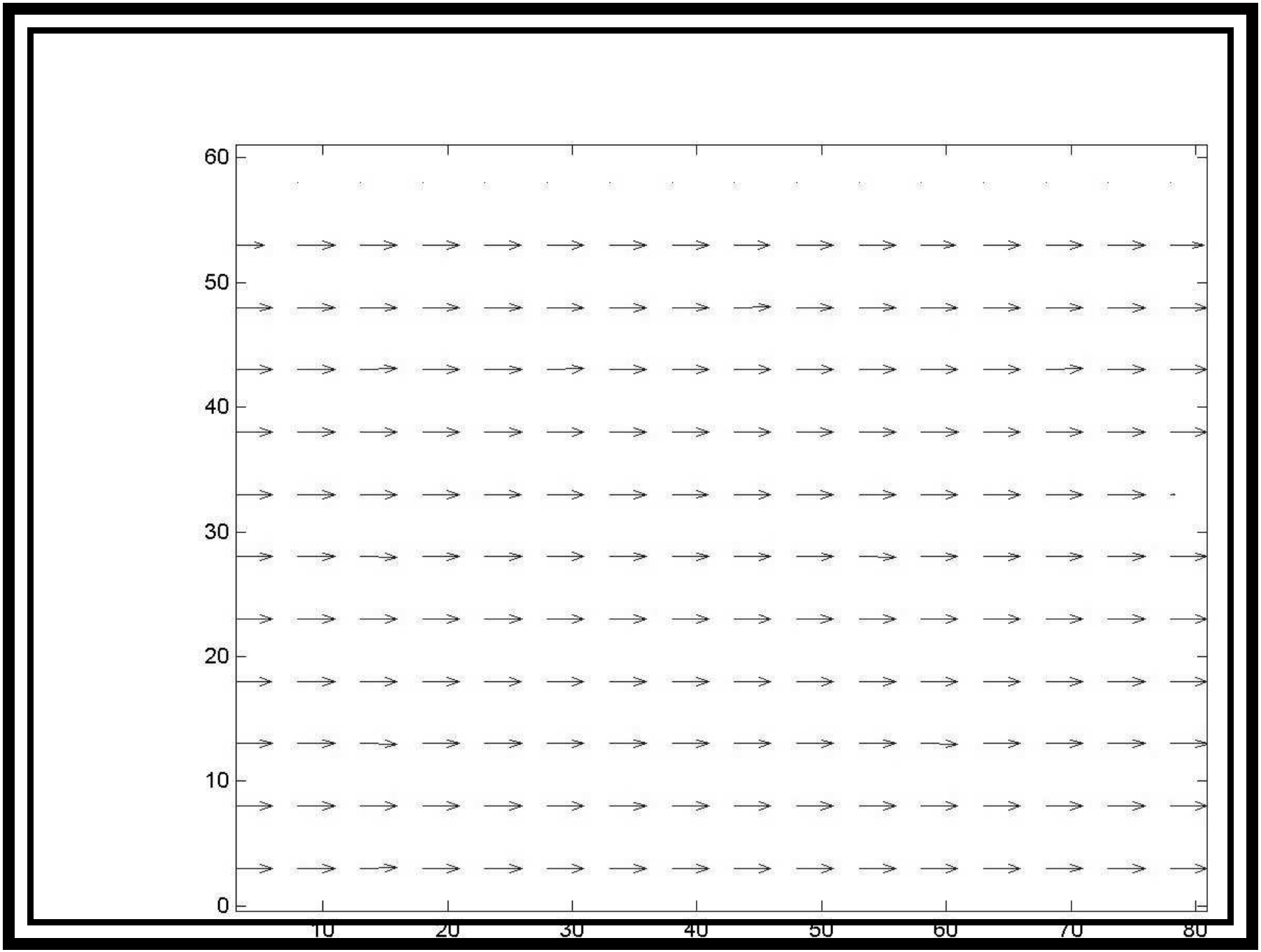
The first term is the image intensity constancy equation and the second term the uniform image motion constraint. The uniform image motion constraints can be approximated by the motion differences between two consecutive frames. The global approach can use the results of the local method as the initializations. The global approach tends to perform better than the local approach, especially near the motion boundaries. Note the formulation is similar to that of the shape from shading.

Optical Flow Estimation Examples

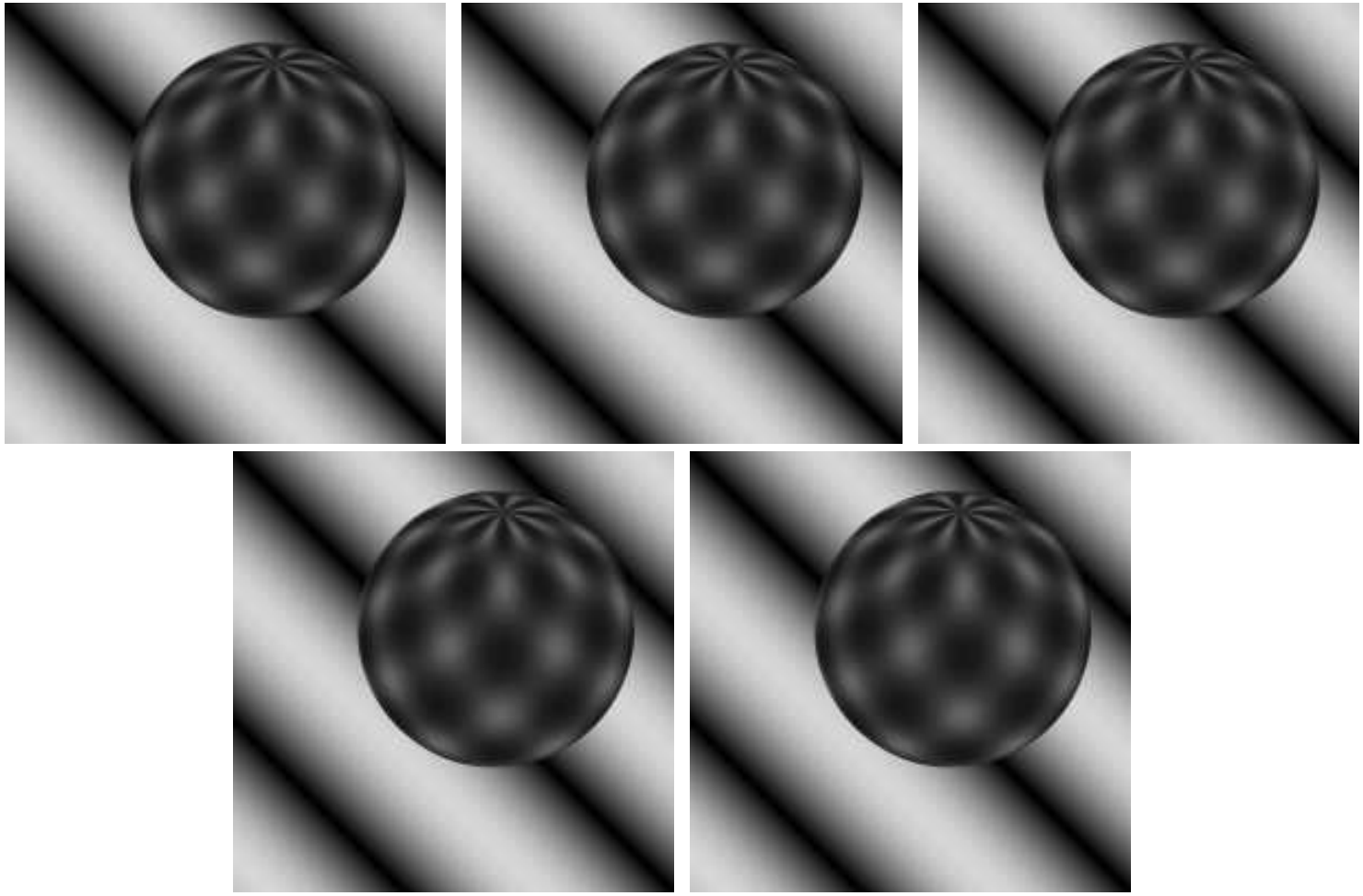
An example of translational movement

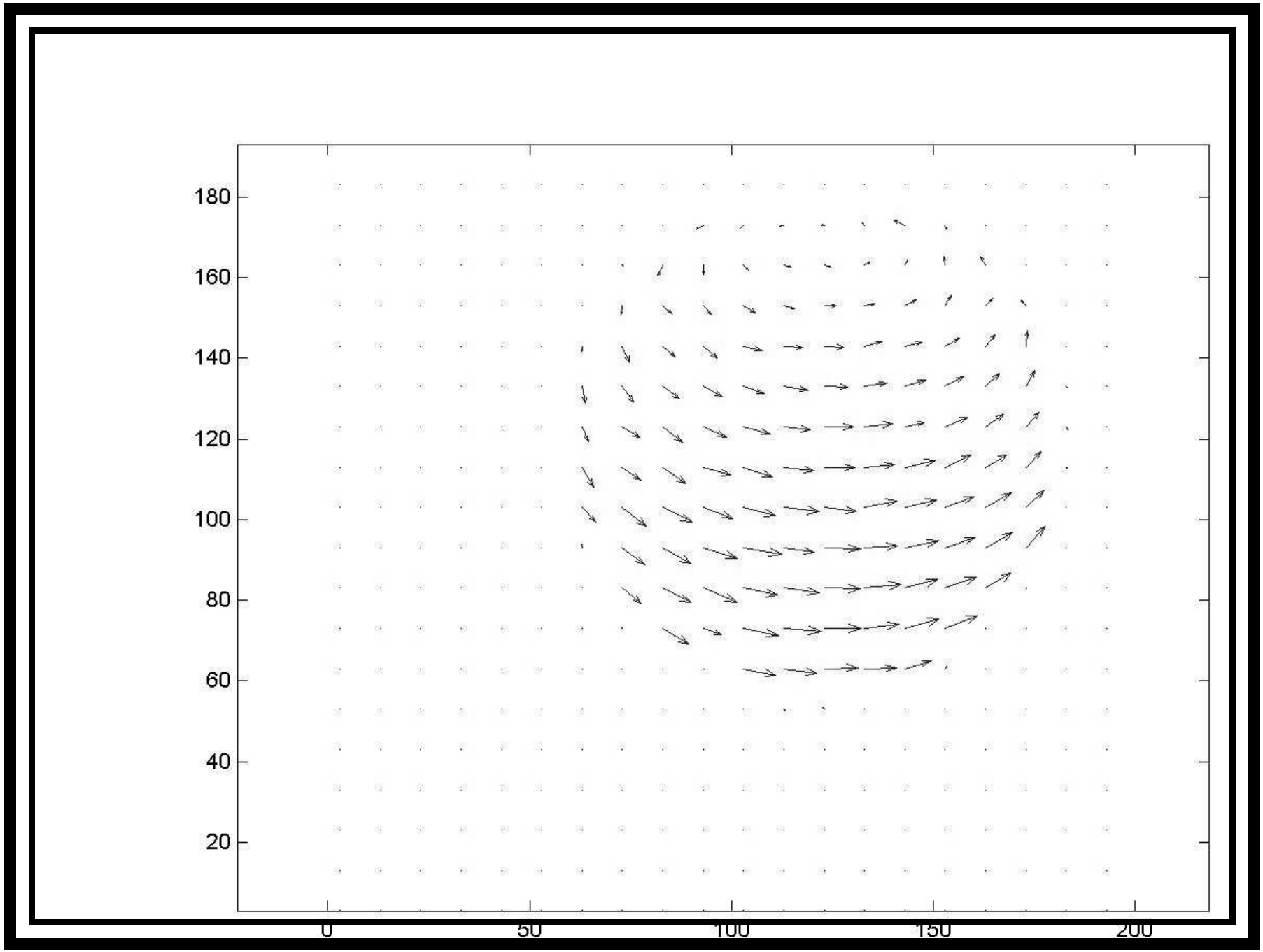





60
50
40
30
20
10
0
10
20
30
40
50
60
70
80

An example of rotational movement




180
160
140
120
100
80
60
40
20
0
50
100
150
200

Motion Analysis from Two Frames

If we are limited to two frames for motion analysis, we can perform motion analysis via a method that combines optical flow estimation with the point matching techniques. The procedure consists of the following steps:

For each small region R in the first image

- Estimate image derivatives numerically (cannot do cubic facet fitting temporally but still can do spatially)
- Estimate optical flow using equation Lucas-Kanade method or the above method
- Produce a new region R' at the next image by *warping* R based on the estimated optical flow. Note warping involves performing a 2D translation and rotation of current point to

a new position based on the estimated optical flow. The orientation of the optical flow determines the rotation while the magnitude of the optical flow determines the translation.

- Compute the correlation between R' and the R in the original image. If the coefficient is large enough, the estimated optical flow is correct. We move on to next location in the image. The result is the optical flow vector for each feature point. Refer to feature point matching algorithm on page 199.

Optical flow methods evaluation and datasets

A good evaluation of different optical flow estimation methods can be found in [6]. Datasets for optical flow estimation and evaluation include the Middlebuary dataset [1], the MPI Sintel dataset [2], and the KITTI dataset [4].

Deep learning methods for OF estimation

For a review of the recent deep learning methods for optical flow estimation, see Chapter 11 of [5].

Sparse Motion Analysis Via Tracking

Tracking is a process that matches a sparse set of feature points from frame to frame.

Kalman Filtering

A popular technique for target tracking is called Kalman filtering. "Filtering" means to obtain the best estimate over time by "filtering out the noise" through estimation fusion. By combining measurements over time, it is a recursive procedure that estimates the next state of a dynamic system, based on the system's current state and it's measurement.

Kalman filtering assumes: 1) linear state model; 2) uncertainty is Gaussian.

Kalman Filtering (cont'd)

Let's look at tracking a point $p = (c_t, r_t)$, where t represents time instant t. Let's the velocity be $v_t = (v_{c,t}, v_{r,t})^T$. Let the state at t be represented by its location and velocity, i.e,
$s_t = [c_t, r_t, v_{c,t}, v_{r,t}]^T$. The goal here is to compute the state vector from frame to frame. More specifically, given s_t, estimate s_{t+1}.

Kalman Filtering (cont'd)

According to the theory of Kalman filtering, s_t, the state vector at the current time frame t, linearly relates to previous state s_{t-1} by the **system model** as follows

$$\mathbf{s}_t = \Phi \mathbf{s}_{t-1} + \mathbf{w_t} \tag{30}$$

where Φ is the state transition matrix and $\mathbf{w_t}$ represents system perturbation, normally distributed as $w_t \sim N(0, Q)$. State model describes the temporal part of the system and it can be probabilistically defined as

$$p(s_t|s_{t-1}) = \mathcal{N}(\Phi s_{t-1}, Q)$$

If we assume the target movement between two consecutive

frames is small enough to consider the motion of target positions from frame to frame uniform, that is,

$$\begin{aligned} c_t &= c_{t-1} + v_{c,t-1} \\ r_t &= r_{t-1} + v_{r,t-1} \\ v_{c,t} &= v_{c,t-1} \\ v_{r,t} &= v_{r,t-1} \end{aligned}$$

Hence, the state transition matrix can be parameterized as

$$\Phi = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The **measurement model** in the form needed by the Kalman

filter is

$$\mathbf{z}_t = H\mathbf{s}_t + \epsilon_t \tag{31}$$

where $z_t = (z_{c_t}, z_{r_t})$ represents the measured target position, measurement matrix H relates current state to current measurement and v_t represents measurement uncertainty, normally distributed as $\epsilon_t \sim N(0, R)$. Measurement model describes the spatial properties of the system and it can be probabilistically defined as

$$p(z_t|s_t) = \mathcal{N}(H s_t, R)$$

For tracking, measurement is obtained via a target detection process.

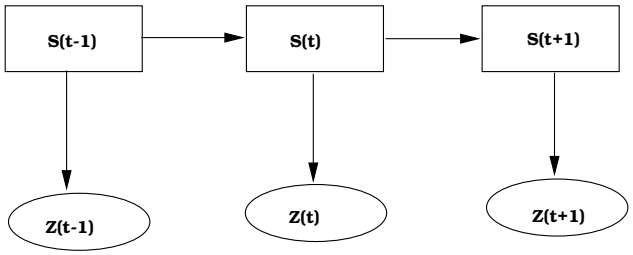
For simplicity and since z_t only involves position [a] and assume z_t is the same as s_t plus some noise, H can be represented as

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

[a] we can also use OF estimation to obtain measurement for the velocity

Kalman Filtering (cont'd)

Kalman filtering consists of recursive state prediction and state updating, i.e., given $p(s_{t-1}|z_{1:t-1}) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$, state prediction is performed using the state model, whereas state updating is performed using the measurement model.



Bayesian network and Markov chain Interpretation for Kalman filtering

Bayesian network: casual relationships between current state and previous state, and between current state and current observation.

Markov process: given S(t), S(t+1) is independent of S(t-1), i.e., S(t+1) is only related to S(t).

Mathematically, it is formulated as given $p(s_{t-1}|z_{1:t-1}) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$, estimate $p(s_t|z_{1:t}) = \mathcal{N}(\mu_t, \Sigma_t)$, i.e., estimating the mean μ_t and the covariance matrix Σ_t of s_t,

the system state at time t.

$$\begin{aligned} p(s_t|z_{1:t}) &= p(s_t|z_{1:t-1}, z_t) \\ &\propto p(s_t|z_{1:t-1})p(z_t|s_t) \end{aligned} \tag{32}$$

where $p(s_t|z_{1:t-1})$ is the state prediction by dynamic model and $p(z_t|s_t)$ is state updating by the measurement model.

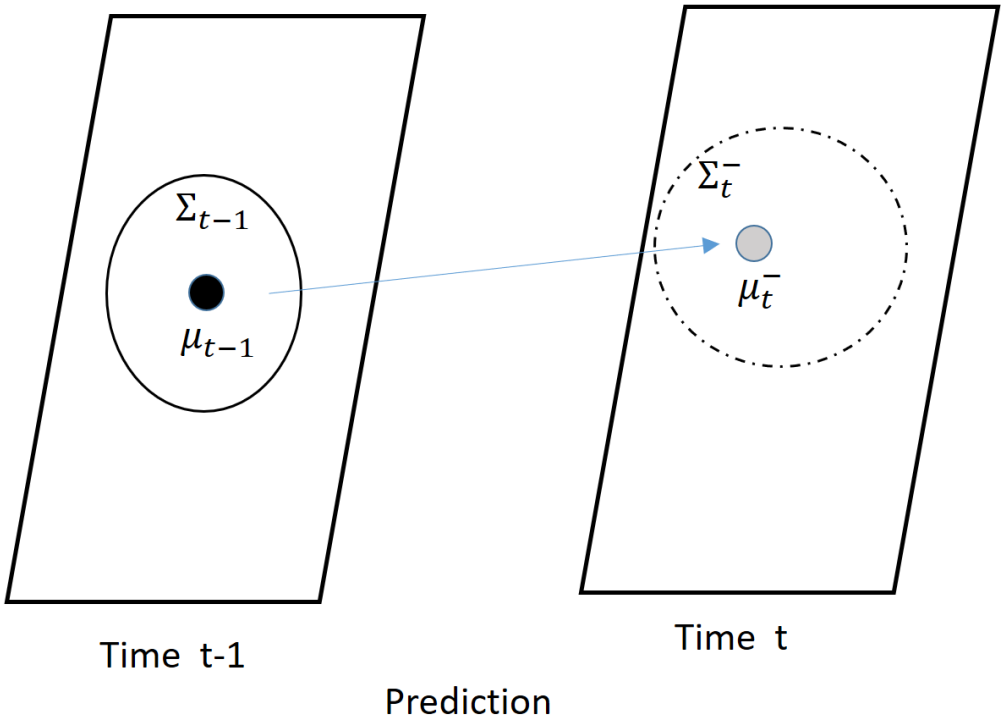
Kalman Filtering (cont'd)

- Prediction- estimate $p(s_t|z_{1:t-1})$, given $p(s_{t-1}|z_{1:t-1}) = \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$,

$$\begin{aligned} p(s_t|z_{1:t-1}) &= \int p(s_t|s_{t-1})p(s_{t-1}|z_{1:t-1})ds_{t-1} \\ &= \int \mathcal{N}(\Phi s_{t-1}, Q)\mathcal{N}(\mu_{t-1}, \Sigma_{t-1})ds_{t-1} \\ &= \mathcal{N}(\mu_t^-, \Sigma_t^-) \end{aligned} \tag{33}$$

  where we can show after some derivations using Eq. 33 that $\mu_t^- = \Phi\mu_{t-1}$ and $\Sigma_t^{-1} = \Phi\Sigma_{t-1}\Phi^T + Q$

  Σ_t^- represents the confidence of the prediction.



- Updating

  Updating uses $p(z_t|s_t) = \mathcal{N}(Hs_t, R)$ to compute $p(s_t|z_{1:t})$ using Eq. 32. It consists of two steps:

  – The first step is the measuring process to obtain

$z_t = (z_{c,t}, z_{r,t})$. The target detector (e.g., thresholding or correlation) searches for the region determined by the covariance matrix Σ_t^- to find the target point at time t. During implementation, the search region, centered at the predicted location, may be a square region of $3\sigma_c \times 3\sigma_r$, where σ_c and σ_r are the two eigen values of the first 2×2 submatrix of Σ_t^-. Correlation method can be used to search the region to identify a location in the region that best matches the detected target in the previous time frame. Search region automatically changes based on Σ_t^-.

– Given z_t, the second step is to combine the prediction (prior) $p(s_t|z_{1:t-1})$ with (likelihood) $p(z_t|s_t)$ using Eq. 32, i.e.,

$$
\begin{aligned}
p(s_t|z_{1:t}) &\propto p(s_t|z_{1:t-1})p(z_t|s_t) \\
&= \mathcal{N}(\mu_t^-, \Sigma_t^-)\mathcal{N}(Hs_t, R) \\
&= \mathcal{N}(\mu_t, \Sigma_t) \qquad (34)
\end{aligned}
$$

By computing the gradient of $\log p(s_t|z_{1:t})$ with respect to s_t and setting it zero, i.e., $\frac{d \log p(s_t|z_{1:t})}{ds_t} = 0$, we can find the mode of $p(s_t|z_{1:t})$, which equals to its mean,

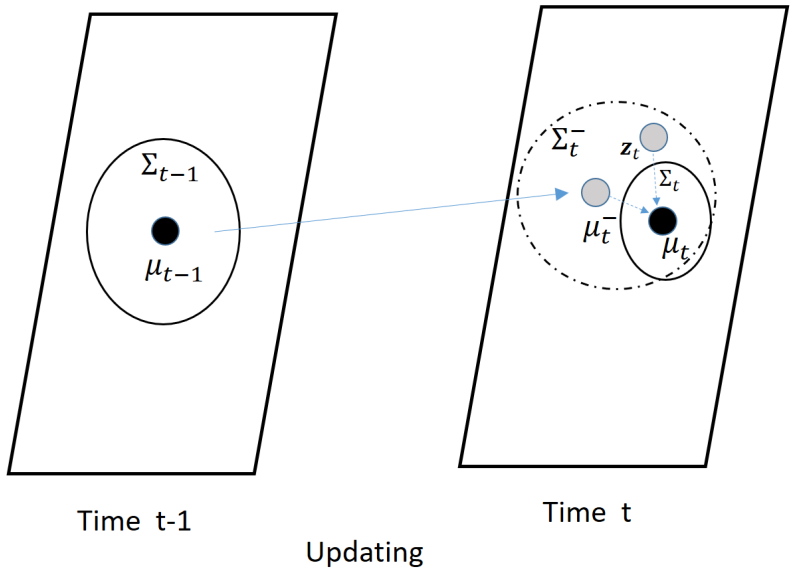
$$
\begin{aligned}
\mu_t &= K_t z_t + (I - K_t H)\mu_t^- \\
&= \mu_t^- + K_t(z_t - H\mu_t^-) \qquad (35)
\end{aligned}
$$

where $K_t = \Sigma_t^- H^T (H\Sigma_t^- H^T + R)^{-1}$. It is the weight (gain) matrix at time t. It weighs the relative contribution of z_t and μ_t^- to s_t. $(z_t - H\mu_t^-)$ is called the innovation.

i.e., the difference between the observed value and the optimal forecast of that value.

Given u_t, Σ^t can be computed as follows

$$\Sigma^t = E[(s_t - \mu_t)(s_t - \mu_t)^T] = (I - K_t H)\Sigma_t^- \qquad (36)$$



Details about the derivations may be found in

https://sites.ecse.rpi.edu/~qji/CV/KF_derivations.pdf

After each time and measurement update pair, the Kalman filter recursively conditions current estimate on all of the past measurements and the process is repeated with the previous *posterior* estimates used to project or predict a new *a priori* estimate. The trace of the state covariance matrix is often used to indicate the uncertainty of the estimated position.

Kalman Filtering Initialization

In order for the Kalman filter to work, the Kalman filter needs be initialized. The Kalman filter is activated after the target is detected in two consecutive frames i and $i+1$

The initial state vector s_0 can be specified as

$$\begin{aligned} c_0 &= c_{i+1} \\ r_0 &= r_{i+1} \\ v_{c,0} &= c_{i+1} - c_i \\ v_{r,0} &= r_{i+1} - r_i \end{aligned}$$

The initial covariance matrix Σ_0 can be given as:

$$\Sigma_0 = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}$$

Σ_0 is usually initialized to very large values. It should decrease and reach a stable state after a few iterations.

We also need initialize the system and measurement error covariance matrices Q and R. The standard deviation from positional system error to be 4 pixels for both c and r directions. We further assume that the standard deviation for velocity error to be 2 pixels/frame. Therefore, the state covariance matrix can

be quantified as

$$Q = \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 16 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

Similarly, we can also assume the error for measurement model as 2 pixels for both c and r direction. Thus,

$$R = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

Both Q and R are assumed be stationary (constant).

Limitations with Kalman Filtering

- assume linear state dynamics (state transition)
- assume the state vector is uni-modal and follows Gaussian distribution, hence cannot track non-Gaussian distributed targets and for Gaussian-distributed targets, it can only track one target at a time.

To overcome these limitations, the conventional Kalman filtering has been extended to *Extended Kalman Filtering* to overcome the linearity assumption, and *Unscented Kalman Filtering* to overcome the Gaussian assumption. For details, see the links to Kalman filtering on course website.

Particle Filtering

A new method based on sampling is called *Particle Filtering* (a.k.a. the condensation algorithm)was introduced to successfully track multi-modal and non-Gaussian distributed objects. The system and observation models are probabilistically generalized to

$$P(s_t|s_{t-1}) \quad \text{system model}$$

$$P(z_t|s_t) \quad \text{observation model}$$

Note that there are no assumption for $P(s_t|s_{t-1})$ and $P(z_t|s_t)$. For Kalman filter, $P(s_t|s_{t-1}) \sim \mathcal{N}(\Phi s_{t-1}, Q)$ and $P(z_t|s_t) \sim \mathcal{N}(H s_t, R)$

Particle Filtering for Tracking

Let s_t represent the state vector and z_t represent the measurement (resulted from a target detector) at time t respectively. Let $z_{1:t} = (z_1, z_2, \ldots, t_t)^T$ be the measurement history and $s_{1:t} = (s_1, s_2, \ldots, s_t)^T$ be the state history. The goal is to determine the posterior probability distribution $P(s_t|z_{1:t})$, from which we will be able to determine a particular (or a set of) s_t, where $p(s_t|z_{1:t})$ are locally maximum.

Particle Filtering for Tracking (cont'd)

$$P(s_t|z_{1:t}) = P(s_t|z_t, z_{t-1}, \ldots, z_1) \quad (37)$$

$$kP(z_t|s_t)P(s_t|z_{1:t-1}) \quad (38)$$

where k is a normalizing constant to ensure the distribution integrates to one. $P(z_t|s_t)$ represents the likelihood of the state s_t and $P(s_t|z_{1:t-1})$ is referred to as *temporal prior*.

$$
\begin{aligned}
P(s_t|z_{1:t-1}) &= \int P(s_t|s_{t-1}, z_{1:t-1}) p(s_{t-1}|z_{1:t-1}) ds_{t-1} \\
&= \int P(s_t|s_{t-1}) P(s_{t-1}|z_{1:t-1}) ds_{t-1} \\
&\approx \frac{1}{N} \sum_{i=1}^{N} P(s_t|s'_{t-1,i}) \qquad (39)
\end{aligned}
$$

where $s'_{t-1,i}$, i=1,2...,N are the samples acquired from sampling $p(s_{t-1}|z_{1:t-1})$. We further assume $z_{1:t-1}$ is independent of s_t given s_{t-1}. So $P(s_t|z_{1:t-1})$ consists of two components: $P(s_t|s_{t-1})$ the temporal dynamics or state transition and $p(s_{t-1}|z_{1:t-1})$ the posterior state distribution at the previous time instant. It shows how the temporal dynamics and the posterior state distribution at the previous time are combined to prodce the state distribution for current time.

The posterior probability of $p(s_t|z_{1:t})$ can be written as

$$\begin{aligned} P(s_t|z_{1:t}) &\approx kP(z_t|s_t)\frac{1}{N}\sum_{i=1}^{N} P(s_t|s'_{t-1,i}) \\ &= \frac{k}{N}\sum_{i=1}^{N} P(z_t|s_t)P(s_t|s'_{t-1,i}) \\ &= \frac{k}{N}\sum_{i=1}^{N} P(z_t|s_{t,i})P(s_{t,i}|s_{t-1}) \\ &= \{s_{t,i}, w_{t,i}\}_{i=1}^{N} \end{aligned} \tag{40}$$

where $w_{i,t} = (z_t|s_{t,i})$. In summary, $P(s_t|z_{1:t})$ is approximated numerically by weighted N samples (particles).

Particle Filtering Pseudo-code

1. At time t, we have $p(s_{t-1}|z_{1:t-1}) = \{s_{t-1,i}, w_{t-1,i}\}$, i=1,2,..,N, where N is the number particles, $t-1, i$ is the ith particle at time $t-1$ and $w_{t-1,i}$ is the weight for the ith particle at time t. At t=0, $s_{0,i}$ is uniformly selected and $w_{0,i} = \frac{1}{N}$.
2. Sample $p(s_{t-1}|z_{1:t-1})$ to acquire samples $s'_{t-1,i}$, i=1,2,...,N. [a]
3. Compute $p(s_t|z_{1:t-1})$ using $s'_{t-1,i}$ according to Eq. 39.
4. Sample $p(s_t|z_{1:t-1})$ to acquire samples $s_{t,i}$, i=1,2,...,N, according to Eq. 39. [b]

[a] $p(s_{t-1}|z_{1:t-1})$ is a mixture of particle distributions; its sampling is performed by first sampling the mixture component index i according to the weight $w_{t-1,i}$, and then obtaining a sample s'_{t-1} from $p(s|s_{t-1,i})$

[b] Sampling $p(s_t|z_{1:t-1})$ consists of two steps: 1)uniformly sample the mixture index i; 2) obtain a sample $s_{t,i}$ from $p(s_t|s'_{t-1,i})$

5. Perform target detection near each particle $s_{t,i}$ to yield measurement $z_{t,i}$

6. Use the measurement model $p(z|s)$ to compute the initial weight $\alpha_{t,i} = p(z_{t,i}|s_{t,i})$ [c]

[c]Note steps 6 and 7 can be combined to directly compute the weight at particle by $\alpha_{t,i} = p(I_{t,i}|s_{t,i})$, where $I_{t,i}$ is the image features extracted from the patch centered at particle i. $p(I_{t,i}|s_{t,i})$ can be computed by comparing $I_{t,i}$ to the target template features.

7. Normalize the initial weights to obtain final weight for each sample $w_{t,i} = \frac{\alpha_{t,i}}{\sum_{i=1}^{N} \alpha_{t,i}}$
8. Obtain particle distribution at time t, $p(s_t|z_{1:t}) = \{s_{t,i}, w_{t,i}\}$
9. Repeat steps 1-8 until t=T, output target distribution$\{s_{t,i}\}_{i=1}^{N}$

Particle Filtering Pseudo-code

- Given $p(s_{t-1}|z_{1:t-1}) = \{s_{t-1,i}, w_{t-1,i}\}_{i=1}^{N}$, first, index each particle as j, j=1,2,.., N, and let $p(j = k) = w_{t-1,k}$. Sample the distribution [a] $\{p(j = 1), p(j = 2), ..., p(j = N)\}$ to obtain a particle index j. Second, given j, sample system model $p(s|s_{t-1,j})$ to obtain $s_{t,i}$, where $p(s|s_{t-1,j}) \sim N(s_{t,j}, Q)$ and Q is system covariance matrix.
- Perform a detection around each $s_{t,i}$ to produce measurements $z_{t,i}$, i=1,2...,N.
- Compute initial weight $\alpha_{t,i}$

  Assume $p(z_t|s_t) \sim N(s_t, R)$, where R is the covariance matrix for measurement perturbation, and then $\alpha_{t,i} = p(z_{t,i}|s_{t,i})$

[a] redistribute the N particles proportionally among the N indexes based on their probabilities, i.e., allocate $N * p(j = k)$ to $s_{t-1,k}$

- Q and R can be manually specified as for Kalman filter.

Particle Filtering

See following videos for further information on particle filtering

Particle filtering lectures

https://www.youtube.com/watch?v=OM5iXrvUr_o

https://www.youtube.com/watch?v=O1FZyWz_yj4

3D Motion and Structure Reconstruction

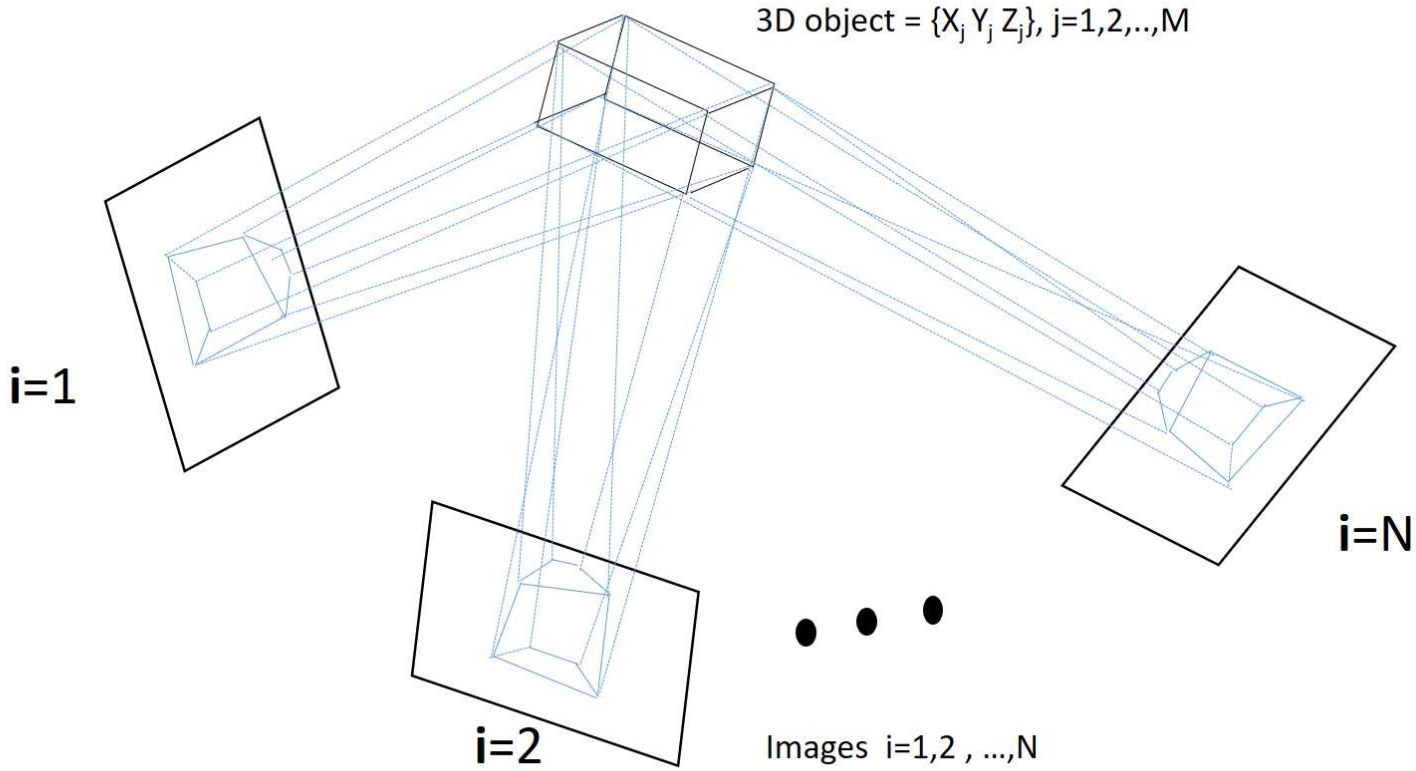
Given the motion field (optical flow) estimated from an image sequence, the goal is to recover the 3D motion and 3D shape of the object. This can be done on the sparse motion fields resulted from Kalman filter or from the dense motion field resulted from optical estimation.

3D Motion and Structure from a Sparse Motion Field 1

We want to reconstruct the 3D structure and motion from the motion field generated by a sparse of target points. Among many methods, we discuss the subspace *factorization method*, i.e., Tomasi-Kanade Factorization.

Factorization Method

Assumptions: 1) the camera is orthographic; 2) M non-coplanar 3D points and N images (N $\geq$ 3), 3) rigid motion (note later work has extended the original factorization to non-rigid factorization.)


3D object = {X_j Y_j Z_j}, j=1,2,..,M
i=1
i=2
i=N
Images i=1,2 , …,N

Factorization Method

Assume the relative rigid motion between the camera and object results from moving the camera while holding the object stationary. Let $p_{ij} = (c_{ij}, r_{ij})$ denote the jth image point on the ith image frame. Let $\bar{c}_i$ and $\bar{r}_i$ be the centroid of the image points on the ith image frame. Let $P_j = (x_j, y_j, z_j)$ be the jth 3D points relative to the object frame and let $\bar{P}$ be the centroid of the 3D points.

The relative coordinates can be computed as

$$\begin{aligned} c'_{ij} &= c_{ij} - \bar{c}_i \\ r'_{ij} &= r_{ij} - \bar{r}_i \\ P'_j &= P_j - \bar{P} \end{aligned}$$

Due to orthographic projection assumption, for the jth point on

the ith image, we have

$$\begin{pmatrix} c'_{ij} \\ r'_{ij} \end{pmatrix} = M_i P'_j$$

$$= \begin{pmatrix} \mathbf{r}_{i,1} \\ \mathbf{r}_{i,2} \end{pmatrix} P'_j$$

where $\mathbf{r}_{i,1}$ and $\mathbf{r}_{i,2}$ are the first two rows of the rotation matrix between camera frame i and the object frame.

By stacking rows and columns, the equations can be written compactly in the form:

$$W = RS$$

where R is $2N \times 3$ matrix and S is $3 \times M$, and

$$R = \begin{pmatrix} \mathbf{r}_{1,1} \\ \mathbf{r}_{1,2} \\ \vdots \\ \mathbf{r}_{N,1} \\ \mathbf{r}_{N,2} \end{pmatrix}$$

$$S = [P'_1 \; P'_2 \ldots, P'_M]$$

Note $\mathbf{r}_{i,1}$ and $\mathbf{r}_{i,2}$ are both row vectors. R gives the relative orientation of each frame to the object frame while S contains the 3D coordinates of the feature points.

$$W^{2N\times M} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1M} \\ r_{11} & r_{12} & \dots & r_{1M} \\ c_{21} & c_{22} & \dots & c_{2M} \\ r_{21} & r_{22} & \dots & r_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ c_{N1} & c_{N2} & \dots & c_{NM} \\ r_{N1} & r_{N2} & \dots & r_{NM} \end{pmatrix}$$

According to the rank theorem,
$Rank(AB) \leq min(Rank(A), Rank(B))$, i.e., the matrix W (often called *registered measurement matrix*) has a maximum rank of 3 (note the rank is not 3 if the motion is non-rigid) . This is evident since the columns of W are linearly dependent on each other due to orthographic projection assumption and the same rotation

matrix for all image points in the same frame. Since $W = RS$, then $rank(W) \leq min(rank(R), rank(S))$. Hence, the maximum rank for W is 3.

Factorization Method (cont'd)

In reality, due to image noise, W may have a rank more than 3. To impose the rank theorem, we can perform a SVD on W

$$W = UDV^T$$

- Change the diagonal elements of D to zeros except for the 3 largest ones.
- Remove the rows and columns of D that do not contain the three largest eigen values, yielding D' of size 3×3. D' is usually the first 3×3 submatrix of D.
- Keep the three columns U that correspond to the three largest singular values of D (they are usually the first three columns) and remove the remaining columns, yielding U', which has a dimension of $2N \times 3$.

- Keep the the three columns V that correspond to the three largest singular values of D (they are usually the first three columns) and remove the remaining columns, yielding V', which has a dimension of $M \times 3$.

$$W' = U' D' V'^T$$

W' is closest to W, yet still satisfies the rank theorem.

Factorization Method (cont'd)

Given W', we can perform decomposition to obtain estimates for S and R. Based on the SVD of $W' = U'D'V'^T$, we have

$$R' = U'D'^{\frac{1}{2}} \qquad\qquad S' = D'^{\frac{1}{2}}V'^T$$

it is apparent $W' = R'S'$

This solution is, however, only up to an affine transformation since for any invertible 3×3 matrix Q, $R = R'Q$ and $S = Q^{-1}S'$ also satisfy the equation. We can find matrix Q using the constraint that from the first row, every successive two rows of R

are orthonormal (see eq. 8.39 of Trucoo's book), i.e.,

$$\begin{aligned} \mathbf{r}_{i,1}\mathbf{r}_{i,1}^T &= 1 \\ \mathbf{r}_{i,2}\mathbf{r}_{i,2}^T &= 1 \\ \mathbf{r}_{i,1}\mathbf{r}_{i,2}^T &= 0 \end{aligned}$$

where $r_{i,1}$ and $r_{i,2}$ are the first and second rows of matrix R and they are row vectors. Since $R = R'Q$, we have $r_{i,1} = r'_{i,1}Q$ and $r_{i,2} = r'_{i,2}Q$, hence

f

$$\begin{aligned} \mathbf{r'}_{i,1}QQ^T\mathbf{r'}_{i,1}^T &= 1 \\ \mathbf{r'}_{i,2}QQ^T\mathbf{r'}_{i,2}^T &= 1 \\ \mathbf{r'}_{i,1}QQ^T\mathbf{r'}_{i,2}^T &= 0 \end{aligned}$$

where $r'_{i,1}$ and $r'_{i,2}$ are the first and second rows of matrix R'.

Given $i = 1, 2, \ldots, N$ frames and using the equations above , we can linearly solve for $A = QQ^T$ subject to the constraint that A is symmetric (i.e., only 6 unknowns). Given A, Q can be obtained via Choleski factorization. Alternatively, we can perform SVD on A, yielding $A = UDU^T$. Hence, $Q = UD^{\frac{1}{2}}$.

Given Q, the final motion estimate is $R = R'Q$ and the final structure estimate is $S = Q^{-1}\hat{S}'$.

Recent work by Kanade and Morris has extended this to other camera models such as affine and weak perspective projection model.

The work by Oliensis (Dec. 2002, PAMI) introduced a new approach for SFM from only two images.

Factorization Method (cont'd)

The steps for structure from motion (SFM) using factorization method can be summarized

- Given the image coordinates of the feature points at different frames, construct the W matrix.
- Compute W' from W using SVD
- Compute R' and S' from W' using SVD
- Solve for the matrix Q linearly.
- $R = R'Q$ and $S = Q^{-1}S'$

see the algorithm on page 208 of Trucco's book.

Factorization Method (cont'd)

Due to the assumption of orthographic assumption, the translation motion is determined as follows. The component of the translation parallel to the image plane is proportional to the frame-to-frame motion of the centroid of the data points on the image plane. The translation component that is parallel to the camera optical axis can not be determined.

Factorization Method for Recovery Non-rigid Motion

see the paper by Christoph Bregler, Araon Hertzmann, and Henning Biermann titled Recovery non-rigid 3d shape from image streams, CVPR00. Implement the algorithm.

also read the paper by Lorenzo Torresani, Aaron Hertzmann,Chris Bregler,

Learning Non-Rigid 3D Shape from 2D Motion, NIPS, 2003.

The algorithm can be downloaded from http://movement.stanford.edu/learning-nr-shape/

3D Motion Estimation from Sparse Motion Field 2

Let $P = (X, Y, Z)$ be a 3D point relative to the camera frame and $p = (c, r)$ be the corresponding image point in the row column frame. Let W be the camera intrinsic matrix and $V = (V_x, V_y, V_z)$ be the 3D motion for P and $v = (v_c, v_r)$ be the corresponding 2D motion for p. Following equation 11, we have

$$\begin{pmatrix} v_c \\ v_r \end{pmatrix} = \frac{W_2 V}{Z} - \frac{\begin{pmatrix} c \\ r \end{pmatrix} V_z}{Z} \tag{41}$$

Eq. 41 provides 2 equations for 4 unknowns-3 for V and 1 for Z.

To produce additional equation, we can assume 1) the 3D motion

and the 3D structure is locally constant; 2) the 3D motion is uniform.

Constant Local Motion and Structure

Let (c, r) be the location of a target and (c', r') be a neighbor of (c, r), i.e., $(c', r') \in N(c, r)$ [a], we have

$$\begin{pmatrix} v_{c'} \\ v_{r'} \end{pmatrix} = \frac{W_2 V}{Z} - \frac{\begin{pmatrix} c' \\ r' \end{pmatrix} V_z}{Z}, \forall (c', r') \in N(c, r)$$

$$Z \begin{pmatrix} v_{c'} \\ v_{r'} \end{pmatrix} = W_2 V - \begin{pmatrix} c' \\ r' \end{pmatrix} V_z$$

[a] This method requires tracking two nearby points.

$$Z \begin{pmatrix} v_{c'} \\ v_{r'} \end{pmatrix} = W_2 V - \begin{pmatrix} 0 & 0 & c' \\ 0 & 0 & r' \end{pmatrix} V = W_2' V$$

where $W_2' = W_2 - \begin{pmatrix} 0 & 0 & c' \\ 0 & 0 & r' \end{pmatrix}$

$$\begin{pmatrix} v_{c'} \\ v_r' \end{pmatrix} Z - W_2' V = 0$$

$$\begin{pmatrix} v_{c'} & -W_{21}' \\ v_r' & -W_{22}' \end{pmatrix} \begin{pmatrix} Z \\ V_x \\ V_y \\ V_z \end{pmatrix} = 0 \qquad (42)$$

where W'_{21} and W'_{22} are the first and second rows of W'_2. Z and V for pixel (c,r) can be computed by minimizing the following loss function

$$
\begin{aligned}
&\left(\begin{array}{cc}
v_{c'}(1) & -W'_{21} \\
v_{r'}(1) & -W'_{22} \\
v_{c'}(2) & -W'_{21} \\
v_{r'}(2) & -W'_{22} \\
\cdots & \\
v_{c'}(N) & -W'_{21} \\
v_{r'}(N) & -W'_{22}
\end{array}\right)
\left(\begin{array}{c} Z \\ V_x \\ V_y \\ V_z \end{array}\right)^T \\
&\underbrace{\left(\begin{array}{cc}
v_{c'}(1) & -W'_{21} \\
v_{r'}(1) & -W'_{22} \\
v_{c'}(2) & -W'_{21} \\
v_{r'}(2) & -W'_{22} \\
\cdots & \\
v_{c'}(N) & -W'_{21} \\
v_{r'}(N) & -W'_{22}
\end{array}\right)}_{A}
\left(\begin{array}{c} Z \\ V_x \\ V_y \\ V_z \end{array}\right) = 0
\end{aligned}
\tag{43}
$$

where $(c'(n), r'(n)) \in N(c, r)$, i.e., the nth neighbor of (c,r).

The solution to $\begin{pmatrix} Z \\ V_x \\ V_y \\ V_z \end{pmatrix}$ is the null vector of the A matrix up to a scale factor.

Uniform 3D motion

For uniform 3D motion, we have $\frac{dV}{dt} = 0$

$$\begin{aligned}
\frac{dv}{dt} &= \frac{d(\frac{W_2 V}{Z})}{dt} - \frac{d(\frac{\begin{pmatrix} c \\ r \end{pmatrix} V_z}{Z})}{dt} \\
&= \frac{W_2}{Z}\frac{dV}{dt} - \frac{W_2 V}{Z^2}\frac{dZ}{dt} - \frac{\begin{pmatrix} c \\ r \end{pmatrix}}{Z}\frac{dV_z}{dt} \\
&- \frac{d\begin{pmatrix} c \\ r \end{pmatrix}}{dt}\frac{V_z}{Z} + \frac{\begin{pmatrix} c \\ r \end{pmatrix} V_z}{Z^2}\frac{dZ}{dt} \\
&= -\frac{W_2 V}{Z^2}V_z - v\frac{V_z}{Z} + \frac{\begin{pmatrix} c \\ r \end{pmatrix} V_z^2}{Z^2} = -2v\frac{V_z}{Z} \qquad (44)
\end{aligned}$$

$$
\begin{aligned}
\frac{dv}{dt} &= -2v\frac{V_z}{Z} \\
Z\frac{dv}{dt} &= -2vV_z \\
Zv' + 2vV_z &= 0 \\
Zv' + 2\begin{pmatrix} 0 & 0 & v \end{pmatrix} V &= 0 \qquad (45)
\end{aligned}
$$

Combining Eq. 42 with Eq. 45 yields

$$
\underbrace{\begin{pmatrix} v_{c'} & -W'_{21} \\ v'_r & -W'_{22} \\ v' & 2\begin{pmatrix} 0 & 0 & v \end{pmatrix} \end{pmatrix}}_{A} \begin{pmatrix} Z \\ V_x \\ V_y \\ V_z \end{pmatrix} = 0 \qquad (46)
$$

From Eq. 46, the solution to $\begin{pmatrix} Z \\ V_x \\ V_y \\ V_z \end{pmatrix}$ is the null vector of A matrix up to a scale factor.

3D Motion and Structure from Dense Motion Field

Given an optical flow field and intrinsic parameters of the viewing camera, recover the 3D motion and structure of the observed scene with respect to the camera reference frame.

3D Motion Parallax

Motion parallax occurs when the images of two 3D moving points coincide momentarily due to projection. The relative motion field of the two instantaneously coincident image points does not depend on the rotational component of motion.

Let two 3D points $P = [X, Y, Z]^T$ and $P' = [X', Y', Z']^T$ be projected into the image points $p = (c, r)$ and $p' = (c', r')$. The corresponding motion vector for point $p = (c, r)$ may be expressed as

$$
\begin{aligned}
v_c &= (c - c_e)\frac{t_z}{Z} - \omega_y f s_x \\
&+ (r - r_0)\omega_z \frac{s_x}{s_y} + \frac{(c - c_0)(r - r_0)\omega_x}{f s_y} - \frac{(c - c_0)^2 \omega_y}{f s_x} \\
&= v_c^T + v_c^\omega \\
v_r &= (r - r_e)\frac{t_z}{Z} + \omega_x f s_y \\
&- (c - c_0)\omega_z \frac{s_y}{s_x} - \frac{(c - c_0)(r - r_0)\omega_y}{f s_x} + \frac{(r - r_0)^2 \omega_x}{f s_y} \\
&= v_r^T + v_r^\omega
\end{aligned} \tag{47}
$$

Similarly, for $p' = (c', r')$, its motion vector is:

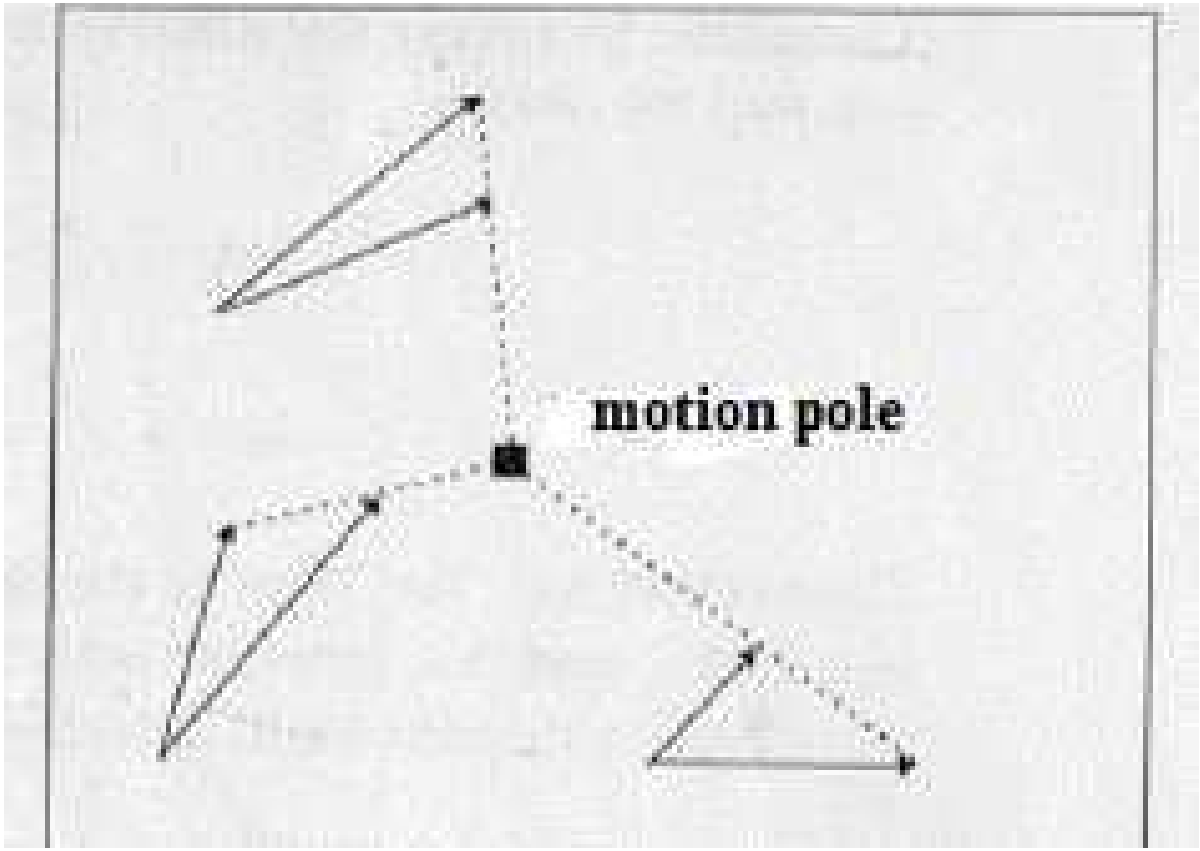
$$
\begin{aligned}
v'_c &= v_c'^T + v_c'^\omega \\
v'_y &= v_r'^T + v_r'^\omega
\end{aligned}
$$

if at some time instant, p and p' are coincident, i.e, $p = p' = (c, r)$, then the relative motion between them can be expressed as the difference of the only translational components as their rotational components are the same ($v^{\omega} = v'^{\omega}$)

$$\begin{aligned} \Delta v_c &= v_c^T - v_c'^T = (c - c_e)(\frac{t_z}{Z} - \frac{t_z}{Z'}) \\ \Delta v_r &= v_r^T - v_r'^T = (r - r_e)(\frac{t_z}{Z} - \frac{t_z}{Z'}) \end{aligned}$$

where (c_e, r_e) defines the motion pole and, $c_e = \frac{f s_x t_X}{t_Z} + c_0$ and $r_e = \frac{f s_y t_Y}{t_Z} + r_0$. It is clear that 1) the relative motion vector $(\Delta v_c, \Delta v_r)$ does not depend on the rotational component of the motion; 2) the relative motion vector points in the direction of $p_e = (c_e, r_e)$ (fig. 8.5) ; 3) $\frac{\Delta v_c}{\Delta v_r} = \frac{c - c_e}{r - r_e}$ (for recovery of the translational motion); 4)

$v_\perp = (v_c, v_r)^T \cdot ((r - r_e), -(c - c_e))^T = v_c^\omega (r - r_e) - v_r^\omega (c - c_e)$ (for recovery of the rotational motion) , where $(r - r_e, -(c - c_e))$ is orthogonal to $(\Delta v_c, \Delta v_r)$. Note $v_c = \frac{t_z}{Z}(c - c_e) + v_c^\omega$ and $v_r = \frac{t_z}{Z}(r - r_e) + v_r^\omega$



Translation Direction Determination

Given two nearby image points p and p' as a result of motion parallax or their spatial closeness, the relative motion field is

$$\begin{aligned} \Delta v_c &= v_c^T - v_c'^T = (c - c_e)(\frac{t_z}{Z} - \frac{t_z}{Z'}) \\ \Delta v_r &= v_r^T - v_r'^T = (r - r_e)(\frac{t_z}{Z} - \frac{t_z}{Z'}) \end{aligned}$$

The second terms on the right side in above equations are negligible if the two points are very close, producing the motion parallax.

Translation Direction Determination (cont'd)

Given a point $p_i = (c_i, r_i)$ and all its close neighbors, $p_j = (c_j, r_j)$, $j \in N_i$, we can compute relative point between p and each of its neighbors. For each relative motion, we have

$$\frac{\Delta v_c}{\Delta v_r} = \frac{c_i - c_e}{r_i - r_e}$$

where (c_i, r_i) is the ith neighbor of p. A least-square framework can be setup to solve for $p_e = (c_e, r_e)$, which, given (c_0, r_0), also leads to the solution to translation motion (t_x, t_y, t_z) up to a scale factor, i,e., $(\alpha t'_x, \alpha t'_y, \alpha t'_z)$- only obtain translation motion direction.

Rotation and Depth Determination

According to Eq. 47, the pointwise dot product between the optical flow (v_{c_i}, v_{r_i}) at point $p_i = (c_i, r_i)$ and the vector $(r_i - r_e, -(c_i - c_e))^T$ yields

$$v_\perp = v^\omega_{c_i}(r_i - r_e) - v^\omega_{r_i}(c_i - c_e) \tag{48}$$

where $v_\perp = (v_{c_i} \quad v_{r_i})^T \cdot (r_i - r_e \quad -(c_i - c_e)^T$ and $v^\omega_{c_i}$ and $v^\omega_{r_i}$ are the rotational component of the 3D motion. They are functions of $(\omega_x, \omega_y, \omega_z)$ as shown in equation 47. Given a set of points i=1,2,..,N, we can have N Eq. 48, which allow to solve for $(\omega_x, \omega_y, \omega_z)$ in a linear least-square framework. Given $(\omega_x, \omega_y, \omega_z)$ and (t_x, t_y, t_z), the depth Z can be recovered using equation 47. From equation 47, we have,

$$v_c = \frac{t_z}{Z}(c - c_e) + v_c^\omega = \frac{\alpha t'_z}{Z}(c - c_e) + v_c^\omega$$
$$v_r = \frac{t_z}{Z}(r - r_e) + v_r^\omega = \frac{\alpha t'_z}{Z}(r - r_e) + v_r^\omega$$

They can be used to solve for Z and the scale factor α linearly by multiplying both sides by Z. This means in the end we can exactly solve for (t_X, t_Y, t_Z), $(\omega_x, \omega_y, \omega_z)$, and Z.

3D Motion based Segmentation

Given a sequence of images taken by a fixed camera, find the regions of the image corresponding to the different moving objects.

3D Motion based Segmentation (cont'd)

Techniques for motion based segmentation

- using optical flow
- image differencing

Optical flow allows to detect different moving objects as well as to infer object moving directions. The optical flow estimates may not be accurate near the boundaries between moving objects.

Image differencing is simple. It, however, can not infer 3D motion of the objects.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[2] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012.

[3] Ravi Garg, Luis Pizarro, Daniel Rueckert, and Lourdes Agapito. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In *Asian Conference on Computer Vision*, pages 460–473. Springer, 2010.

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[5] Joel Janai, Fatma Güney, Aseem Behl, Andreas Geiger, et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art.

Foundations and Trends® in Computer Graphics and Vision, 12(1–3):1–308, 2020.

[6] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.