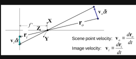
Motion Field



Scene point velocity $v_o = \frac{dr_o}{dt}$

Perspective projection: $\frac{1}{f} r_i = \frac{r_o}{r_o \cdot Z}$

Image velocity $v_i = \frac{dr_i}{dt}$

Motion field

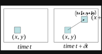
$$v_i = \frac{dr_i}{dt} = f' \frac{(r_o \cdot Z) v_o - (v_o \cdot Z) r_o}{(r_o \cdot Z)^2} = f' \frac{(r_o \times v_o) \times Z}{(r_o \cdot Z)^2}$$

Optical Flow

To estimate pixel motion from image H to image I, we take a pixel in H and look for nearby pixels of the same color in I.

Assumptions:
- Color constancy: a point in H looks the same in image I. For grayscale, this is brightness constancy
- Small motion: points do not move very far

Optical flow is determined only for pixels.



Optical flow velocities (u,v)

Displacement: $(\delta x, \delta y) = (u\delta t, v\delta t)$

Assume brightness of patch remains same in both images:

$$E(x+u\delta t, y+v\delta t, t+\delta t) = E(x,y,t)$$

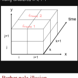
$$E(x,y,t) + \delta x \frac{\partial E}{\partial x} + \delta y \frac{\partial E}{\partial y} + \delta t \frac{\partial E}{\partial t} = E(x,y,t)$$

$$\lim_{\delta t \to 0} \frac{\partial E}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial E}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial E}{\partial t}\frac{\partial t}{\partial t} = 0$$

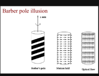
$$u\frac{\partial E}{\partial x} + v\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0$$

Constraint equation: $E_x u + E_y v + E_t = 0$ (u,v) cannot be found uniquely with just this constraint

Finding Gradients in X-Y-T



$$E_x = \frac{1}{4\delta x}\Big[\big(E_{i+1,j,k} + E_{i+1,j,k+1} + E_{i+1,j+1,k} + E_{i+1,j+1,k+1}\big) - \big(E_{i,j,k} + E_{i,j,k+1} + E_{i,j+1,k} + E_{i,j+1,k+1}\big)\Big]$$



Optical flow constraint ambiguity
Here we get direction only in vertical gradient, not the other gradient.

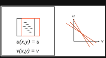
Aperture problem is an example too of ambiguity.

There is also a problem in determining u and v. They depend not only on the object motion, but also its distance from the camera. Farther objects appear to move slower - motion parallax.

Even if we assume translation of all the points in the scene, we still have N+3 unknowns because of the Z.

Constant Flow Case

We will assume that in a small area, multiple pixels will have the same flow. In such a case, u and v are the same for the pixels in the same area.



From the optical flow constraint equation, we find that its u and v lies on a line. Now for the next pixels, we get more lines from their constraint equation and since they have the same u and v, they must intersect.

$E = \min_{u,v} \sum_{x,y \in W} (uI_x + vI_y + I_t)^2$ we estimate the intersection by minimizing this equation

$$\frac{\partial E}{\partial u} = 0 = 2\sum (uI_x + vI_y + I_t) I_x$$

$$0 = \sum (uI_x^2 + vI_yI_x + I_tI_x)$$

$$\frac{\partial E}{\partial v} = 0 = \sum (uI_xI_y + vI_y^2 + I_yI_t)$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum I_xI_t \\ \sum I_tI_y \end{bmatrix}$$

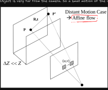
We saw this in Harris corner detection

u and v can be estimated by this matrix is invertible. If there is no texture, this inverse would not exist.
So, u and v can be estimated if there is a texture.

Low texture region -- corner detection would not work because gradients have small magnitude -- u and v cannot be estimated properly
Edges (aperture problem) -- wont work because of all same and large gradients
High textured region -- gradients are different and have large magnitudes -- u and v can be estimated locally, patch by patch wherever there is texture.

Distant Motion Case: Affine Flow

Object is very far from the camera. So a small motion of the image would translate to a large skewed motion of the object.



Affine flow

$u(x,y) = ax + by + c$

$v(x,y) = dx + ey + f$

Instead of assuming u and v as constants, they are now a function of x and y.

With this assumption, an object can be tracked if it is moving far from the camera. It still does not solve motion parallax.

$$\min_{a,b,c,d,e,f} \sum_{(x,y) \in W} \left[(ax+by+c)I_x + (dx+ey+f)I_y + I_t\right]^2$$

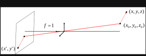
On solving, we get

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = -\begin{bmatrix} \sum x^2I_x^2 & \sum xyI_x^2 & \sum xI_x^2 & \sum x^2I_xI_y & \sum xyI_xI_y & \sum xI_xI_y \\ & \sum y^2I_x^2 & \sum yI_x^2 & \sum xyI_xI_y & \sum y^2I_xI_y & \sum yI_xI_y \\ & & \sum I_x^2 & \sum xI_xI_y & \sum yI_xI_y & \sum I_xI_y \\ & & & \sum x^2I_y^2 & \sum xyI_y^2 & \sum xI_y^2 \\ & & & & \sum y^2I_y^2 & \sum yI_y^2 \\ & & & & & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum xI_xI_t \\ \sum yI_xI_t \\ \sum I_xI_t \\ \sum xI_yI_t \\ \sum yI_yI_t \\ \sum I_yI_t \end{bmatrix}$$

Focus of Expansion

Motion of object = - (Motion of Sensor)

For a given translatory motion and gaze direction, the world seems to flow out of one point (FOE)



For translation only:
- All the flow vector intersect at one point the FoE (Focus of Expansion) where there is zero motion
- The flow is greater further from the FOE

After time t, the scene point moves to

$$(x, y, z) = (x_0 + ut, y_0 + vt, z_0 + wt)$$

$$(x', y') = \left(f\frac{x_0 + ut}{z_0 + wt}, f\frac{y_0 + vt}{z_0 + wt}\right)$$

To find FOE, backtrack to time $t \to -\infty$

$$\lim_{t \to -\infty} (x', y') = \left(f\frac{x_0 + ut}{z_0 + wt}, f\frac{y_0 + vt}{z_0 + wt}\right) = \left(\frac{fu}{w}, \frac{fv}{w}\right)$$

Lucas Kanade Image Alignment and Tracking

Warp and its Jacobian

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

If there's a translation, the warp would estimate how the image would look after translation.

$$W = \begin{bmatrix} 1 & 0 & p_1 \\ 0 & 1 & p_2 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine warp

$$W = \begin{bmatrix} p_1 & p_3 & p_5 \\ p_2 & p_4 & p_6 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Jacobian

$$\frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}_{2 \times N}$$

Jacobian of affine warp $W_x = p_1x + p_3y + p_5$, $W_y = p_2x + p_4y + p_6$, $W(x;p)$

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

We want to warp an image (I(W(x;p))) to minimize

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

T(x) is the template

$\min_p \sum_x [I(W(x;p)) - T(x)]^2$ pixel to pixel difference in a particular neighborhood

This is generally a nonlinear optimization problem. We solve for increments to current estimate.

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

$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$

We need to warp the image and then compute the derivative

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

$$\frac{\partial E}{\partial \Delta p} = 0 = 2\sum_x \left[\nabla I \frac{\partial W}{\partial p}\right]^T \left[I(W(x;p)) + \nabla I \frac{\partial W}{\partial p}\Delta p - T(x)\right]$$

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

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

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

Lucas Kanade Algorithm:

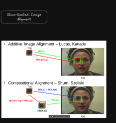
1. Warp image I with W(x;p) to get I(W(x;p))
2. Compute error image T(x) - I(W(x;p))
3. Warp gradient of I to compute ∇I
4. Evaluate Jacobian $\frac{\partial W}{\partial p}$
5. Compute Hessian
6. Compute Δp
7. Update parameters $p \leftarrow p + \Delta p$

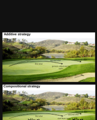
$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \qquad \frac{\partial W}{\partial p} = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \cdots & \frac{\partial W_x}{\partial p_N} \\ \frac{\partial W_y}{\partial p_1} & \cdots & \frac{\partial W_y}{\partial p_N} \end{bmatrix}_{2 \times N}$$

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

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

Shum-Szeliski Image Alignment





Compositional Warp

Minimize

$$\sum_x [I(W(W(x;\Delta p);p)) - T(x)]^2 \approx \sum_x \left[I(W(W(x;0);p)) + \nabla I(W)\frac{\partial W}{\partial p}\Delta p - T(x)\right]^2$$

This Jacobian is evaluated at (x;0). Hence it is constant.

This Taylor series is expanded around 0, not p

$$I(W(W(x;0+\Delta p));p) = I(W(W(x;0);p)) + \nabla I \frac{\partial W}{\partial p}\Delta p$$
$$= I(W(x;p)) + \nabla I \frac{\partial W}{\partial p}\Delta p$$

$W(x;p) \circ W(x;\Delta p)$

$$= \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix}\begin{bmatrix} 1+\Delta p_1 & \Delta p_3 & \Delta p_5 \\ \Delta p_2 & 1+\Delta p_4 & \Delta p_6 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

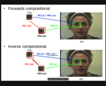
$W(x;p+\Delta p)$

$$= \begin{bmatrix} p_1+\Delta p_1 & p_3+\Delta p_3 & p_5+\Delta p_5 \\ p_2+\Delta p_2 & p_4+\Delta p_4 & p_6+\Delta p_6 \end{bmatrix}$$

Compositional Warp Algorithm:

1. Warp image I with W(x;p) to get I(W(x;p))
2. Compute error image T(x) - I(W(x;p))
3. Warp gradient of I to compute ∇I
4. Evaluate Jacobian $\frac{\partial W}{\partial p}$. This is constant
5. Compute Hessian
6. Compute Δp
7. Update $W(x;p) \leftarrow W(x;p) \circ W(x;\Delta p)$

Inverse Compositional Warp



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

$$\approx \sum_x \left[T(W(x;0)) + \nabla T \frac{\partial W}{\partial p}\Delta p - I(W(x;p))\right]^2$$

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

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

- Jacobian is constant - evaluated at (x, 0)
- Gradient of template is constant
- Hessian is constant
- Can pre-compute everything but error image

Update
$W(x;p) \leftarrow W(x;p) \circ W(x;\Delta p)^{-1}$

Inverse Compositional Warp Algorithm:

1. Warp image I with W(x;p) to get I(W(x;p))
2. Compute error image T(x) - I(W(x;p))
3. Warp gradient of I to compute ∇I
4. Evaluate Jacobian $\frac{\partial W}{\partial p}$. This is constant
5. Compute Hessian. This is constant
6. Compute Δp. ∇T while computing this is constant
7. Update $W(x;p) \leftarrow W(x;p) \circ W(x;\Delta p)^{-1}$

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

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

Algorithm	Can be applied to	Efficient?	Authors
Forwards Additive	Any	No	Lucas, Kanade
Forwards Compositional	Any semi-group	No	Shum, Szeliski
Inverse Compositional	**Any group**	Yes	Baker, Matthews
Inverse Additive	*Simple linear 2D+*	Yes	*Hager, Belhumeur*