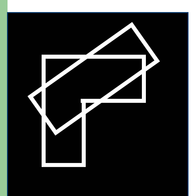
Generalized Hough Transform

Line Fitting



$$x\cos\theta + y\sin\theta = \rho$$

Circle Fitting

- Use the gradient direction θ at the edge point

- Compute x_0, y_0 given x, y for fixed intervals of r

$$x_0 = x - r\cos\theta$$
$$y_0 = y - r\sin\theta$$

Generalized Hough Transform

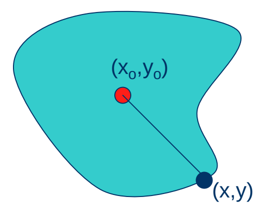
- For shapes with ***no*** analytical expression
- Requires learning of r-table
  - Distance vector and its angle for each boundary pixel to object centroid
- Finding a shape in an image
  - For input edge-map compute angle (can be from image gradient)
  - For each distance vector corresponding to edge angle increment a 2D accumulator array corresponding to (x_0, y_0)

Generating R-table

- Compute centroid
- For each edge compute its distance to centroid

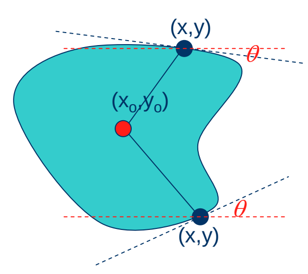
$$r = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

- Find edge orientation
- Construct a table of angles and *r* values



Generating R-table

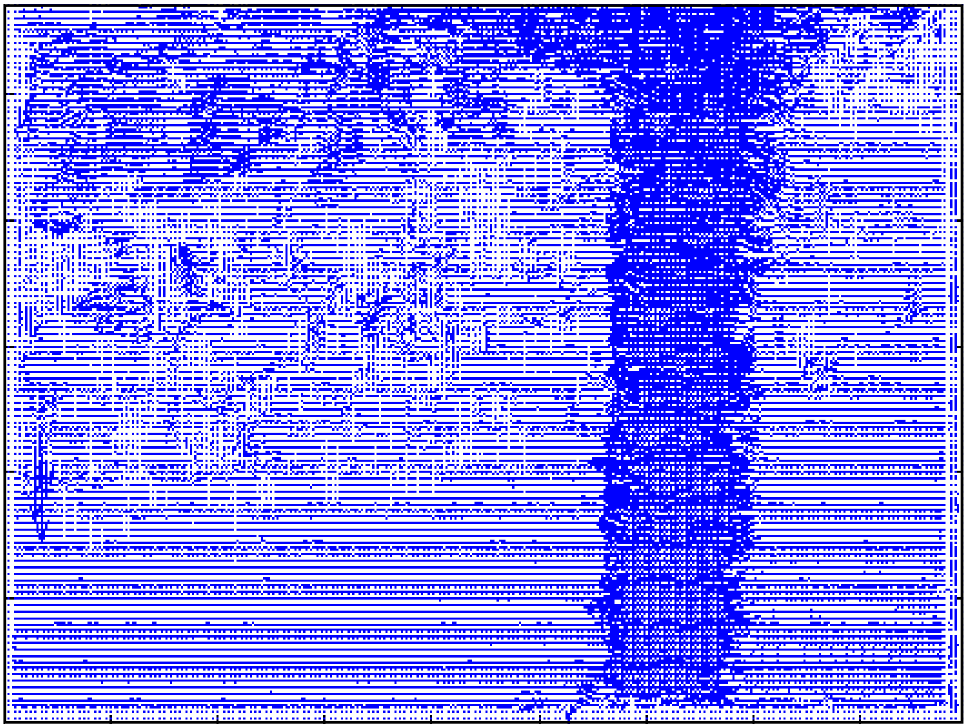
θ1	r1, r2, r3 …
θ2	r14, r21, r23 …
θ3	r41, r42, r33 …
θ4	r10, r12, r13 …



Detecting Shape

- Knowns
  - Edge points (x,y)
  - Gradient angle at every edge point θ
  - R-table of shape need to be detected
- For each edge point find θ go to corresponding row of R-table
- Create an accumulator array ***A*** of 2D (x,y)
  - Increment ***A*** using the *r* values.

Optical Flow



Optical Flow

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

$$0 = \frac{\Delta x}{\Delta t} I_x + \frac{\Delta y}{\Delta t} I_y + I_t$$

$$u = \frac{\Delta x}{\Delta t} \qquad v = \frac{\Delta y}{\Delta t}$$

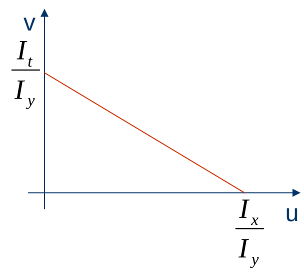
$$v = u \frac{I_x}{I_y} + \frac{I_t}{I_y}$$

Equation of a line in (*u*,*v*) space

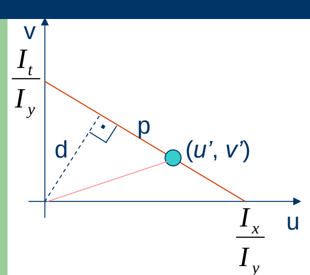
Optical Flow

- We know I_x, I_y, and I_t from images
- We have 2 unknowns: u, v
- Solution lie any where on the line

$$v = u\frac{I_x}{I_y} + \frac{I_t}{I_y}$$



Optical Flow



$$v = u\frac{I_x}{I_y} + \frac{I_t}{I_y}$$

- Let (u', v') be true flow
- True flow has two components
  - Normal flow: d
  - Parallel flow: p
- Normal flow **can be** computed
- Parallel flow **cannot**

Computing True Flow

- Horn & Schunck
- Schunk
- Lukas & Kanade

Horn & Schunck

$$uI_x + vI_y + I_t = 0$$

- Define an energy function and minimize

$$E(x,y) = (uI_x + vI_y + I_t)^2 + \lambda \overbrace{(u_x^2 + u_y^2 + v_x^2 + v_y^2)}^{f}$$

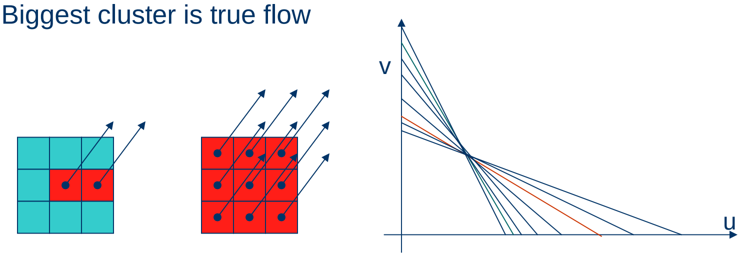
$$u\left(\lambda + I_x^2\right) + vI_xI_y + I_xI_t - \lambda u_{avg} = 0$$

$$v\left(\lambda + I_y^2\right) + uI_xI_y + I_yI_t - \lambda v_{avg} = 0$$

$$u = u_{avg} - I_x\left(\frac{I_xu_{avg} + I_yv_{avg} + I_t}{I_x^2 + I_y^2 + \lambda}\right) \qquad v = v_{avg} - I_y\left(\frac{I_xu_{avg} + I_yv_{avg} + I_t}{I_x^2 + I_y^2 + \lambda}\right)$$

Schunck

- If two neighboring pixels move with same velocity
  - Corresponding flow equations intersect at a point in (u,v) space
  - Find the intersection point of lines
  - If more than 1 intersection points find clusters
  - Biggest cluster is true flow



Lucas & Kanade

- Define an energy functional
  - Take derivatives equate it to 0
  - Rearrange and construct an observation matrix

$$E = \sum (uI_x + vI_y + I_t)^2$$

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

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

$$\overbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} u \\ v \end{bmatrix}}^{u} = \overbrace{\begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}}^{B}$$

Lucas & Kanade

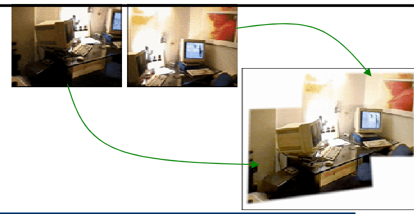
$$Au = B \qquad A^{-1}Au = A^{-1}B \qquad Iu = A^{-1}B \qquad u = A^{-1}B$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum I_x^2 \sum I_y^2 - \left(\sum I_x I_y\right)^2} \begin{bmatrix} \sum I_y^2 & -\sum I_x I_y \\ -\sum I_x I_y & \sum I_x^2 \end{bmatrix} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

Global Flow

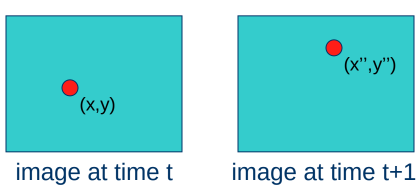
Global Motion



- Common motion of pixels observed in frame
  - Camera motion or rigid object motion
- Affine Model
  - Affine Transformation
  - Affine Motion

Affine Transformation

- Direct relation between pixel positions



$$x'' = a_1 x + a_2 y + b_1$$
$$y'' = a_3 x + a_4 y + b_2$$

Affine Motion

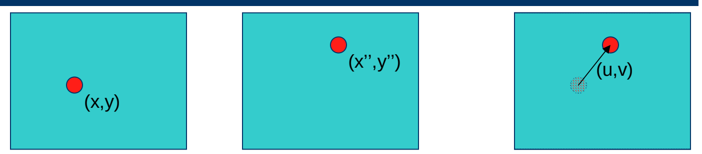


image at time t   image at time t+1

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$u = x'' - x$$
$$x'' - x = a_1 x + a_2 y + b_1$$
$$\boxed{x'' = (a_1 + 1)x + a_2 y + b_1}$$

$$v(x,y) = a_3 x + a_4 y + b_2$$
$$v = y'' - y$$
$$\boxed{y'' = a_3 x + (a_4 + 1)y + b_2}$$

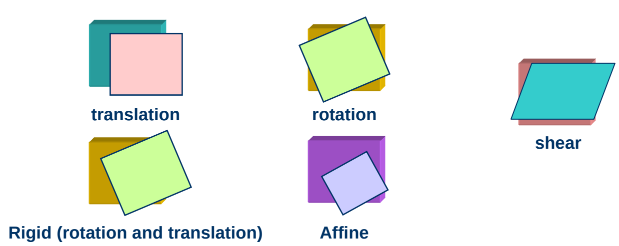
Affine Motion and Transformation

- Transformation

$$x'' = a_1 x + a_2 y + b_1$$
$$y'' = a_3 x + a_4 y + b_2$$

Motion

$$x'' = (a_1 + 1)x + a_2 y + b_1$$
$$y'' = a_3 x + (a_4 + 1)y + b_2$$



James R. Bergen, P. Anandan, Keith J. Hanna, Rajesh Hingorani: "Hierarchical Model-Based Motion Estimation," ECCV 1992: 237-252

Anandan's Approach (Affine Motion)

$$u(x,y) = a_1 x + a_2 y + b_1$$
$$v(x,y) = a_3 x + a_4 y + b_2$$

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ a_3 \\ a_4 \\ b_2 \end{bmatrix}$$

$$\mathbf{u} = \mathbf{X}\mathbf{a}$$
$$\Delta I^T \mathbf{u} + I_t = 0$$
$$\Delta I^T \mathbf{X}\mathbf{a} + I_t = 0$$

minimize $E = \sum \left(\Delta I^T \mathbf{X}\mathbf{a} + I_t \right)^2$

$$\frac{\partial E}{\partial \mathbf{a}} = 2 \sum (\Delta I^T \mathbf{X})^T (I_t + \Delta I^T \mathbf{X}\mathbf{a}) = 0$$

$$\sum_{\text{all pixels}} \mathbf{X}^T \Delta I \Delta I^T \mathbf{X}\mathbf{a} = - \sum_{\text{all pixels}} \mathbf{X}^T \Delta I \; I_t$$