lec 8

optical flow
Video Analysis

① optical flow → detect Motion between Sequential frame, motion of objects, surface, edges

② optical flow ≡ disparity

③ Applications
- Motion based Segmentation
- Structure from motion (3d)
- Video Compression
- Alignment (Global motion Compensation)
  ↳ vide Stabilization, UAV video Analysis

④ Moment of video → Slow → no distortion in optical flow
→ high → distortion, de-formation

⑤ horn / Schunk optical flow (Brightness Constancy Assumption)

$f(x,y,t) = f(x+dx, y+dy, t+dt)$ التغير في الزمن

taylor series $= f(x,y,t) + \frac{df}{dx}dx + \frac{df}{dy}dy + \frac{df}{dt}dt$

$= f_x dx + f_y dy + f_t dt = 0$ ← $\frac{df}{dt} = f_t$

$\div dt$

$f_x u + f_y v + f_t = 0$

$v = \frac{-f_x}{f_y}u - \frac{f_t}{f_y}$

↑ eq of Straight line

P = Parallel flow

d = normal flow

↳ $= \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$

/ /

⑥ horn / Schunck optical flow

look at it as <u>optimization problem</u>

For each pixel ⟶ equation as [Variational] Calculus

a) Brightness Constancy ⟶ small

b) Smoothness Constraint ⟶ small

minumum , maximum ⟶ Calculate derivative of equation (4) (V) then = 0

$$\iint \underbrace{(f_x u + f_y v + f_t)^2}_{\text{Brightness Constancy}} + \lambda \underbrace{(u_x^2 + u_y^2 + v_x^2 + v_y^2)}_{\text{Smoothness Constrain}} \, dx\,dy$$

$$(f_x u + f_y v + f_t) f_x + \lambda (\Delta^2 u) = 0$$

$$(f_x u + f_y v + f_t) f_y + \lambda (\Delta^2 v) = 0$$

$\Delta^2 u = u_{xx} + u_{yy}$ (Second der.)

Laplacian of Gaussian ممكن اعوض

Diffrance of Gaussian عنه ب

Change in directions

x: $u = u_{av} - f_x \frac{P}{D}$

y: $V = V_{av} \; f_y \, P/D$

$P = f_x u_{av} + f_y v_{av} + f_t$

$d = \lambda + f_x^2 + f_y^2$

<u>Algorithm horn/schunck</u>

① k=0 ② initilize

③ Repeat until some error measure is Satisfied (Converge)

/ /

⑦ Lucas Kanade optical flow (least squares)

$f_x u + f_y v = -f_t$ → اخلى ال (f_t) هو اللى طرف لوحدها

Consider [3X3] window

$f_{x_1} u + f_{y_1} v = -f_{t_1}$ ...... $x, y, t = 3$

$AU = f_t$

f_x, f_y — u, v

$$U = (A^t A)^{-1} A^t f_t$$

$$\min \sum (f_{x_i} u + f_{y_i} v + f_t)^2$$

least square fit

{ * Lucas Kanade without pyramids
⇒ fail in areas of large motions }

* horn, schunck, lucas kanade work only for small motion

* if object move faster, (brightness) changes (rapidly) 2X2, 3X3 makes (fail) to estimate spatiotemporal derevative

* (Pyramids) can be used to compute large optical flow vector

Pyramids

useful in representing images
built using multiple copies of images
each pyramid level is (1/4) size of previous level
lowest level → highest resolution
highest level → lowest resolution

Gaussian, laplacian Pyramids

Reduce ↑ expand

level 1

① Gaussian pyramid (reduce)

$$g_l(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n)\, g_{l-1}(2i+m, 2j+n)$$

5×5 mask mask

-2, -1, 0, 1, 2

quality reduce to half (x, y)

total = مجموع

$$g_l = \text{REDUCE}[g_{l-1}]$$

Separability of Gaussian (Reduce 1D)

Gaussian on 2d XX

Gaussian x → Gaussian y

② Gaussian Pyramids (Expand)

$$g_{l,n}(i,j) = \sum_{p=-2}^{2} \sum_{q=-2}^{2} w(p,q)\, g_{l,n-1}\left(\frac{i-p}{n}, \frac{j-q}{n}\right)$$

$$G_{l,n} = \text{Expand}[g_{l,n-1}]$$

Convolution Mask

$$[w(-2), w(-1), w(0), w(1), w(2)]$$

separable: $w(m,n) = \hat{w}(m)\,\hat{w}(n)$

Symetric: $\hat{w}(i) = \hat{w}(-i)$

$$[c, b, a, b, c] \equiv [-2, -1, 0, 1, 2]$$

Sum of mask = 1 $\quad a + 2b + 2c = 1$

All nodes at given level must contribute the same total weight to nodes at next higher level

$$a + 2c = 2b \longrightarrow$$

احمل (X) اول وبالديم النا تج مع (y) ← مع بعض X و y → بدل ما اكمل Gassian

قولنالا قبل كده ↑

Separability of Gaussian pyramid

① Apply 1D Mask to pixels along each row

② Apply 1D Mask to pixels along each Coloumn
↳ from result image from ①

③ Laplacian Pyramid
⟹ Simillar to edge detection
most pixels are (Zero) → 2 Sequential Frames
Can be used for image Compression

Code

$L_1 = g_1 - expand[g_2]$
$L_2 = g_2 - expand[g_3]$
$L_3 = g_3 - expand[g_4]$
$L_4 = g_4$

De Code

$g_4 = L_4$
$g_3 = L_3 + expand\ g_4$
$g_2 = L_2 + expand\ g_3$
$(g_1) = L_1 + expand\ g_2$
↳ reConstructed Image

Laplacian Pyramid → Image Combining

① Generate Laplacian pyramid for 1st img (L1)
② " " " " 2nd img (L2)
③ Generate L3 by:
Copy left half of 1st img
Copy right half of 2nd img
④ ReConstruct Combined img from (L3)

/ /

Lucas Kande with pyramid

- Compute LK optical flow at highest level
- take flow → u_{i-1}, v_{i-1} from level $i-1$
- bilinear Interpolate to create u_i^*, uv^*
- multiply u_i^*, v_i^* by 2
- Compute (P-t) → $u_i^*(x,y)$ $v_i^*(x,y)$
  - Apply LK to get $u_i'(x,y)$ $v_i'(x,y)$ (Correction flow)
  - Add Correction $u_i' v_i'$

$\Rightarrow u_i = u_i' + u_i^*$ $\quad v_i = v_i' + v_i^*$