Video Processing

Lab 2: Lucas-Kanade

Administration

- Contact:
    - vptau2022@gmail.com
    - Course Forum
- HW solutions: We would like you to be <u>critical</u> regarding the results you obtain.

Optical Flow - Problem statement

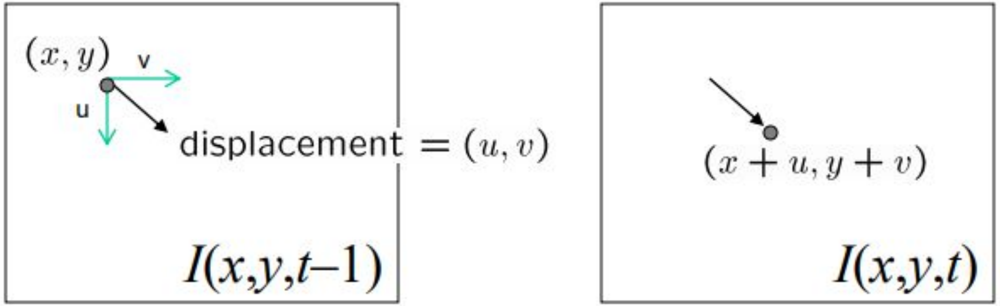
Given:

I(x,y,t-1) I(x,y,t)

I.shape = HxWx3

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

Find: (u, v)



u.shape = HxW
v.shape = HxW

Optical Flow - Mathematical Recap.

Input:

Two images:

1. $I(x, y, t-1) \in R^{h \times w \times 3}$
2. $I(x, y, t) = I(x + u(x, y), y + v(x, y), t) \in R^{h \times w \times 3}$

Output:

Two maps:

1. $u(x, y) \in R^{h \times w}$
2. $v(x, y) \in R^{h \times w}$

Optical Flow - Mathematical Recap.

Math:

First order approximation of Taylor expansion:

$$I(x+u, y+v, t) \approx I(x, y, t-1) + I_x \cdot u(x, y) + I_y \cdot v(x, y) + I_t \cdot 1$$

$$\underbrace{I(x+u, y+v, t) - I(x, y, t-1)}_{\text{LHS}} \approx +I_x \cdot u(x, y) + I_y \cdot v(x, y) + I_t$$

And we have that the Left Hand Side:

$$LHS = I(x+u, y+v, t) - I(x, y, t-1) = 0$$

So, overall:

$$0 = I_x \cdot u(x, y) + I_y \cdot v(x, y) + I_t$$

Optical Flow - Mathematical Recap.

So, overall:

$$0 = I_x \cdot u(x, y) + I_y \cdot v(x, y) + I_t$$

Or otherwise:

$0 = \nabla I \cdot [u, v]^T + I_t \rightarrow$ this is an equation per pixel.

$0 = \nabla I \cdot [u, v]^T + I_t \rightarrow$ this is an equation per pixel.

Optical Flow - Mathematical Recap.

The problem:

We have $H \cdot W$ equations and $2 \cdot H \cdot W$ unknowns.

The solution:

Pretend the pixel's neighbors have the same (u,v): If we use a 5x5 window, that gives us 25 equations per pixel!

$$\begin{bmatrix} I_x(pixel_1) & I_y(pixel_1) \\ I_x(pixel_2) & I_y(pixel_2) \\ . & . \\ . & . \\ . & . \\ I_x(pixel_{25}) & I_y(pixel_{25}) \end{bmatrix}_{25\times2} \begin{pmatrix} u \\ v \end{pmatrix}_{2\times1} = -\begin{pmatrix} I_t(pixel_1) \\ I_t(pixel_2) \\ . \\ . \\ . \\ I_t(pixel_{25}) \end{pmatrix}_{25\times1}$$

$0 = \nabla I \cdot [u, v]^T + I_t$ → this is an equation per pixel.

Optical Flow - Mathematical Recap.

The solution:

Pretend the pixel's neighbors have the same (u,v): If we use a 5x5 window, that gives us 25 equations per pixel!

$$\begin{bmatrix} I_x(pixel_1) & I_y(pixel_1) \\ I_x(pixel_2) & I_y(pixel_2) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ I_x(pixel_{25}) & I_y(pixel_{25}) \end{bmatrix}_{25\times 2} \begin{pmatrix} u \\ v \end{pmatrix}_{2\times 1} = -\begin{pmatrix} I_t(pixel_1) \\ I_t(pixel_2) \\ \cdot \\ \cdot \\ \cdot \\ I_t(pixel_{25}) \end{pmatrix}_{25\times 1}$$

We write it as:

$$A_{25x2} d_{2x1} = b_{25x1}$$

And solve it as a Least-Squares problem: $minimize : ||Ad - b||^2$

$(A^T A)d = A^T b$ and finally: $d = (A^T A)^{-1} \cdot A^T \cdot b$

So far, when do we break?

Where does our math breaks:

1. Brightness constancy is **not** satisfied
2. The motion is **not** small
3. A point does **not** move like its neighbors

In class you saw

In class you saw:

Lucas-Kanade is a parametric method. That is, we can assume that the movement model is :

1. Translation: $W(x;p) = \begin{pmatrix} x + P_1 \\ y + P_2 \end{pmatrix}$
2. Affine: $W(x;p) = \begin{pmatrix} P_1 \cdot x + P_2 \cdot y + P_3 \\ P_1 \cdot x + P_2 \cdot y + P_4 \end{pmatrix}$
3. Whatever you want to model.

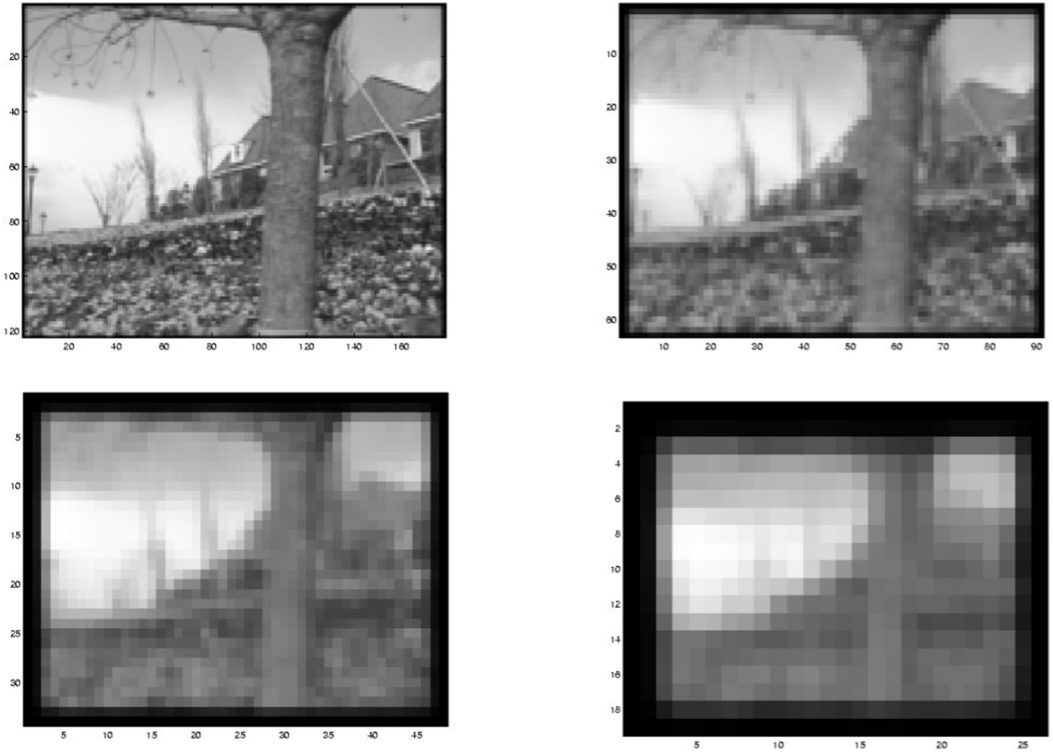
The goal of LK: $p^* = argmin_p \sum [I_2(W(x;p)) - I_1(x)]^2$

where: x = observed area and: p = warp parameters.

How do we solve the assumption that (u, v) are small?

Image pyramid.

Image pyramid



Back to: $p^* = argmin_p \sum [I_2(W(x;p)) - I_1(x)]^2$

We would like to solve this equation:

1. Over levels of the pyramid
2. In an iterative manner.

$$\Delta p^* = argmin_{\Delta p} \sum [I_2(W(x; p + \Delta p) - I_1(x)]^2 =$$

$$argmin_{\Delta p} \sum [I_2(W(x; p) + \nabla I_2 \cdot \Delta p - I_1(x)]^2 =$$

$$argmin_{\Delta p} \sum [I_t + \nabla I_2 \cdot \Delta p]^2$$

This boils down, again, to solving:

$$\nabla I_2 \cdot \Delta p = -I_t$$

Back to: $p^* = argmin_p \sum [I_2(W(x;p)) - I_1(x)]^2$

$\nabla I_2 \cdot \Delta p = -I_t$

which is (according to the class notations):

$B \cdot \Delta p = -I_t$

Where: $\Delta p = \begin{pmatrix} \Delta P_1 \\ \Delta P_2 \end{pmatrix} \in R^{2\times1}$

$$B = \nabla I = \begin{bmatrix} I_x(c_1) & I_y(c_1) \\ I_x(c_2) & I_y(c_2) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ I_x(c_{25}) & I_y(c_{25}) \end{bmatrix} \in R^{25\times2}$$

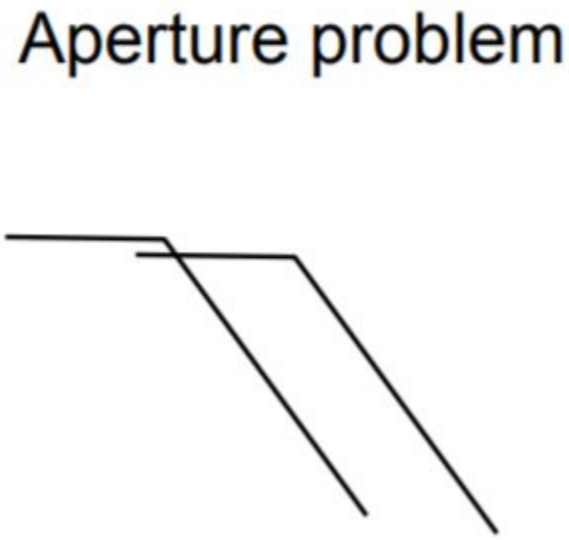
The Least-Squares solution is:

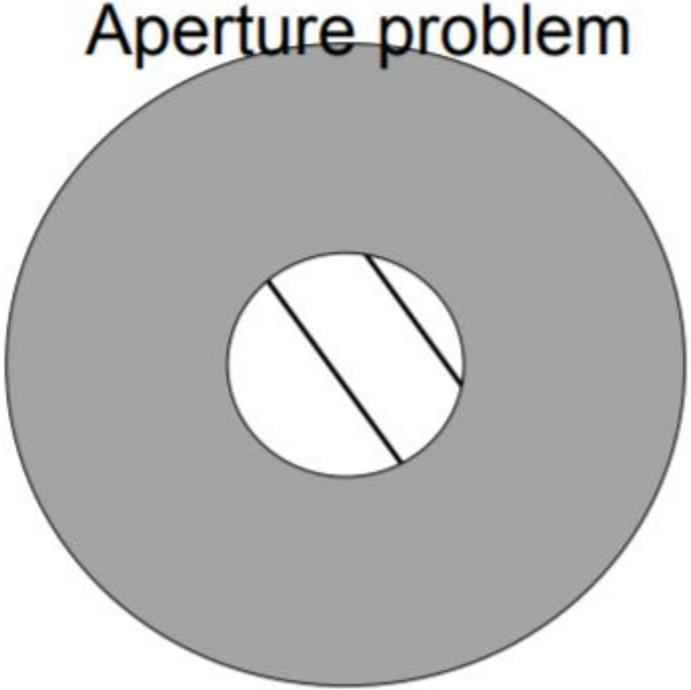
$\Delta p = -(B^T B)^{-1} \cdot B^T \cdot I_t$

And the fill algorithm

1. Compute pyramids for I_1 and I_2.
2. For level in num_of_pyramid_levels:
    1. Warp I_2 using image in the current level and the current (u, v) parameters: I_{2-warp} = WarpImage(Pyr2[level],u ,v)
    2. For step in range(max_iterations):
        1. du, dv = LucasKanadeSingleIteration(Pyr1[level], I_{2-warp}, window_size)
        2. u = u + du
        3. v = v + dv
        4. I_{2-warp} = WarpImage(Pyr2[level], u, v)
    3. Adjust u, v according to the level (multiply them by $\times 2$)

The Aperture Problem





solve using corners

Lets code