

CSE185

Introduction to Computer Vision

Lab 10: Optical Flow

Instructor: Prof. Ming-Hsuan Yang
TA: Taihong Xiao & Tiantian Wang

Input 1



Input 2

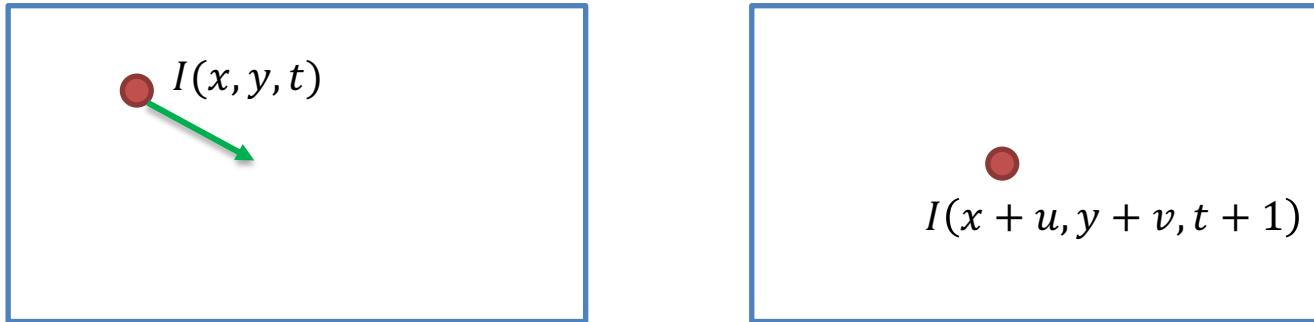


Optical Flow



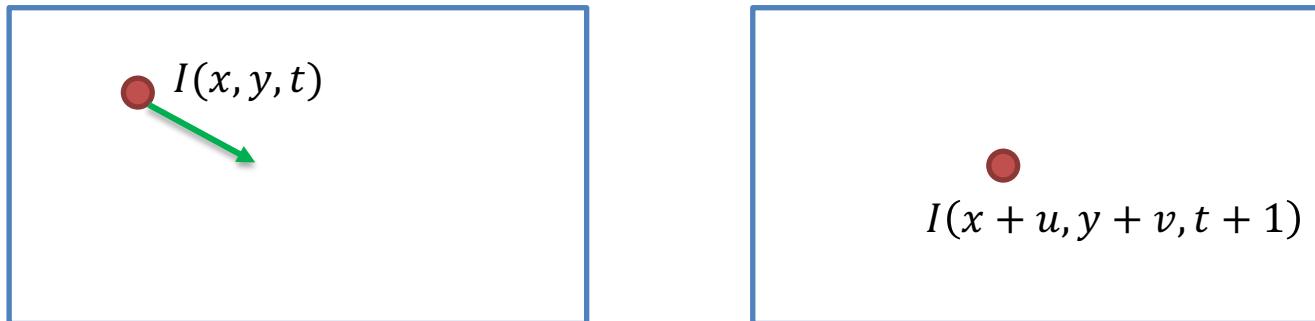
Brightness Consistency

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



Brightness Consistency

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



- Tayler expansion:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v + I_t$$
$$I(x + u, y + v, t + 1) - I(x, y, t) = I_x \cdot u + I_y \cdot v + I_t = 0$$

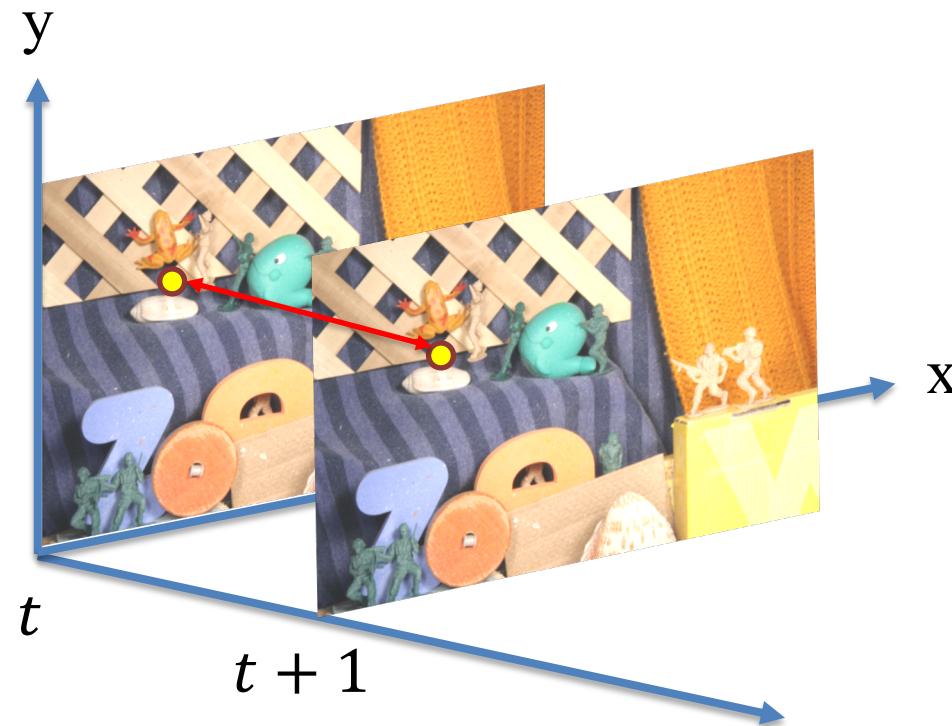
- Brightness consistency equation:

$$I_x \cdot u + I_y \cdot v + I_t = 0$$

- I_x : x-gradient
- I_y : y-gradient
- I_t : pixel difference on time domain

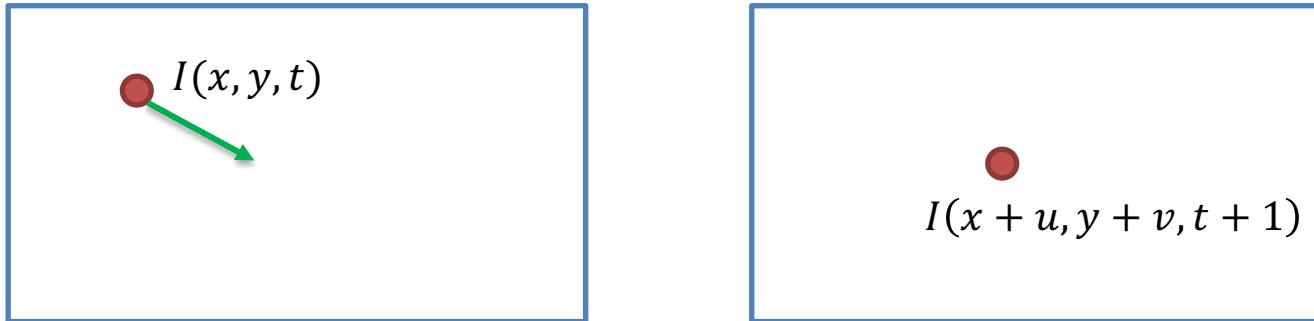
Gradients

- $I_x = I(x + 1, y, t) - I(x, y, t)$
- $I_y = I(x, y + 1, t) - I(x, y, t)$
- $I_t = I(x, y, t + 1) - I(x, y, t)$



Brightness Consistency

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



- We want to solve (u, v) such that:

$$I_x \cdot u + I_y \cdot v = -I_t$$

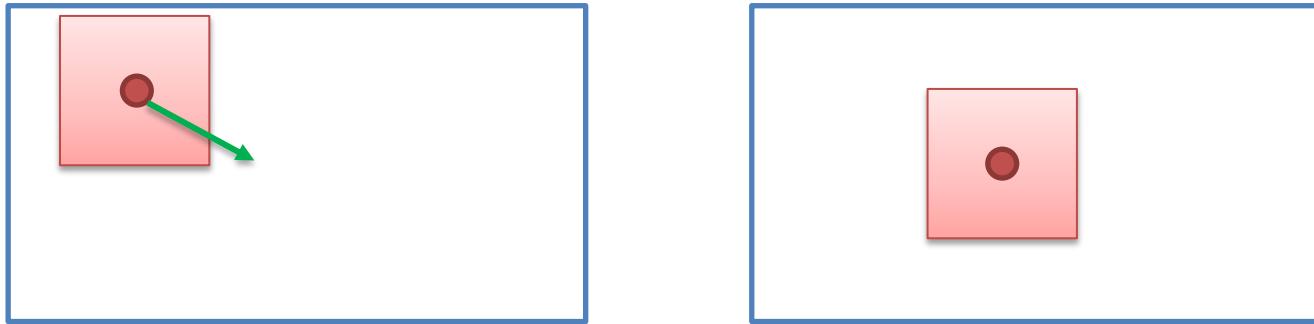
1 equation,
2 variables

- Matrix-vector form:

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} = -I_t$$

Spatial Coherence

- Assume neighbors have the same (u, v)



- A $w \times w$ window gives us w^2 equations:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{w^2}) & I_y(p_{w^2}) \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{w^2}) \end{bmatrix}$$

Solve Linear Equation

- Solve $Ax = b$ in MATLAB:
 - left-division: $x = A \setminus b;$
 - pseudo inverse: $x = \text{pinv}(A) * b;$
 - least square solution: $x = \text{inv}(A' * A) * A' * b;$

Solve Linear Equation

- Solve $Ax = b$ in MATLAB:
 - left-division: $x = A \setminus b;$
 - pseudo inverse: $x = \text{pinv}(A) * b;$
 - least square solution: $x = \text{inv}(A' * A) * A' * b;$
- Solve optical flow equation:

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{w^2}) & I_y(p_{w^2}) \end{bmatrix}}_A \cdot \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_x = -\underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{w^2}) \end{bmatrix}}_b$$

Algorithm

- Pseudo code:

```
Input: I1, I2, window size w
```

```
Output: flow vector (u, v) for each pixel
```

```
u = 0, v = 0 for every pixel
```

```
for each pixel in I1:
```

```
    compute Ix, Iy, It from w x w window
```

```
    convert Ix, Iy, It to vectors
```

```
    let A = [Ix, Iy], b = -It
```

```
    solve x
```

```
    u = x(1), v = x(2)
```

```
end
```

Iterative Refinement

- Pseudo code:

Input: I_1, I_2 , window size w

Output: flow vector (u, v) for each pixel

$u = 0, v = 0$ for every pixel

Run k times:

for each pixel in I_1 :

 compute I_x, I_y from $w \times w$ window

 shift window of I_2 by (u, v) , compute I_t

 convert I_x, I_y, I_t to vectors

 let $A = [I_x, I_y]$, $b = -I_t$

 solve x

$u += x(1), v += x(2)$

end

end

Algorithm

- lab10.m:

```
window_size = 45;  
k = 4;  
  
w = floor(window_size/2);  
shift = w + 10;  
  
I1 = rgb2gray(img1);  
I2 = rgb2gray(img2);  
  
Ix_m = imfilter(I1, [1 -1; 1 -1], 'replicate');  
Iy_m = imfilter(I1, [1 1; -1 -1], 'replicate');  
  
u = zeros(size(I1)); u_next = zeros(size(I1));  
v = zeros(size(I1)); v_next = zeros(size(I1));
```

pre-compute Ix and Iy

Algorithm

- lab10.m:

```
for t = 1 : k
    for i = 1 + shift : size(Ix_m, 1) - shift
        for j = 1 + shift : size(Ix_m, 2) - shift
            %% extract Ix, Iy, It from local window

            %% convert Ix, Iy, It to vectors

            %% construct matrix A and vector b

            %% solve A x = b
            x = [0, 0]; % remove this line
            u_next(i, j) = x(1);
            v_next(i, j) = x(2);
        end
    end
    %% update flow
    u = u + u_next;
    v = v + v_next;
end
```

Hints

- Extract I_x , and I_y from local window:

```
window_size = 45;  
w = floor(window_size/2);  
  
Ix = Ix_m(i-w : i+w, j-w : j+w);
```

the same as you did
in spatial filtering

- Shift the window of I_2 from (i, j) to $(i+v, j+u)$ when extract I_t from local window:

```
i2 = i + v(i, j);  
j2 = j + u(i, j);  
  
It = I1(i - w : i + w, j - w : j + w)  
    - I2(i2 - w : i2 + w, j2 - w : j2 + w);
```

Hints

- Your I_x and I_y are $w \times w$ matrixs, first convert them to $w^2 \times 1$ vectors, and concatenate them into matrix A

```
A = [Ix, Iy];
```

- A is a $w^2 \times 2$ matrix, b is a $w^2 \times 1$ vector, solve x by
 - left-division: $x = A \setminus b;$
 - pseudo inverse: $x = \text{pinv}(A) * b;$
 - least square solution: $x = \text{inv}(A' * A) * A' * b;$

Visualization

- `plot_flow(img2, u, v)`

Iteration 1



Visualization

- `plot_flow(img2, u, v)`

Iteration 2



Visualization

- `plot_flow(img2, u, v)`

Iteration 3



Visualization

- `plot_flow(img2, u, v)`

Iteration 4



TODO

- Finish lab10.m
- Try different window size and different image pairs
- Upload lab10.m and at least 3 results