Optical Flow Estimation

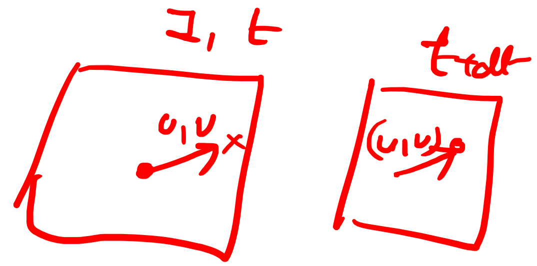
1 Horn & Schunck

2. Lucas & Kanade

3. Pyramids based

Optical flow →

(u,v)



1. Horn and Schunk Approach

B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, Aug. 1981.

Horn and Schunck Optical Flow

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

- True optical flow will satisfy these constraints:
- 1. Brightness Constraint Assumption
- 2. Smoothness constraint Assumption

nearby pixel tend flow in same dir.

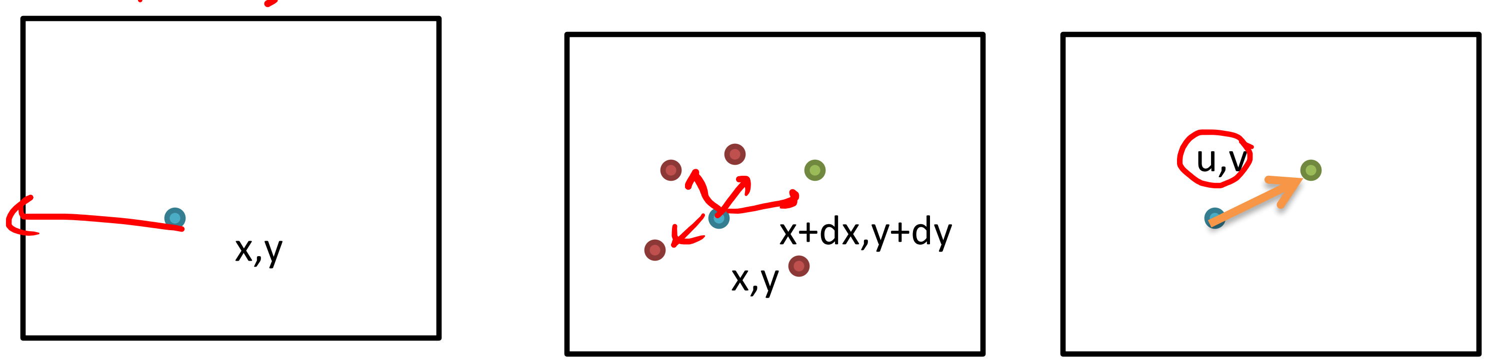
$$\boxed{f_x u + f_y v + f_t = 0}$$

? ?

$f_x = I_x$

$f_y =$

$f(x,y)$



Time=t Time=t+dt Time=t+dt

Horn and Schunck Optical Flow

Smoothness (Const.) The gradient mag. of the flow feild is small

- Estimate u,v , such that the error function E is minimized:

BC SM

$$E(x,y) = (f_x u + f_y v + f_t)^2 + \lambda({u_x}^2 + {u_y}^2 + {v_x}^2 + {v_y}^2)$$

- It can also be written as

$$E_{\mathrm{HS}}(u,v) = \sum_{x,y} \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right)^2 + \lambda \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right)$$

$$= E_{\mathrm{data}}(u,v) + \lambda E_{\mathrm{smoothness}}(u,v)$$

where instead of f, we represents image using I, and summation has to be performed over all pixels of Image

Horn and Schunck Optical Flow

- Estimate u,v , such that the error function E is mininized:

$$E(x,y) = (f_x u + f_y v + f_t)^2 + \lambda({u_x}^2 + {u_y}^2 + {v_x}^2 + {v_y}^2)$$

- It can also be written as

$$E_{\mathrm{HS}}(u,v) = \sum_{x,y} \left(\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} \right)^2 + \lambda \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 \right)$$

$$= E_{\mathrm{data}}(u,v) + \lambda E_{\mathrm{smoothness}}(u,v)$$

where instead of f, we represents image using I

- where λ is a parameter that specifies the influence of the smoothness term, also known as a **regularization** parameter.
- The larger the value of λ, the smoother the optical flow field

Horn and Schunck Optical Flow

- The partial derivatives of the spatiotemporal function *I* are approximated using finite differences between the two given images. That is, at pixel *(x,y)*,

$t \quad t+1$

$I_1, \; I_2$

$$f_x = \frac{\partial I}{\partial x} \approx \frac{1}{4}\big(I_1(x+1,y) - I_1(x,y) + I_1(x+1,y+1) - I_1(x,y+1) + I_2(x+1,y) - I_2(x,y) + I_2(x+1,y+1) - I_2(x,y+1)\big)$$

$$f_y = \frac{\partial I}{\partial y} \approx \frac{1}{4}\big(I_1(x,y+1) - I_1(x,y) + I_1(x+1,y+1) - I_1(x+1,y) + I_2(x,y+1) - I_2(x,y) + I_2(x+1,y+1) - I_2(x+1,y)\big)$$

$$f_t = \frac{\partial I}{\partial t} \approx \frac{1}{4}\big(I_2(x,y) - I_1(x,y) + I_2(x+1,y) - I_1(x+1,y) + I_2(x,y+1) - I_1(x,y+1) + I_2(x+1,y+1) - I_1(x+1,y+1)\big)$$

Horn and Schunck Optical Flow

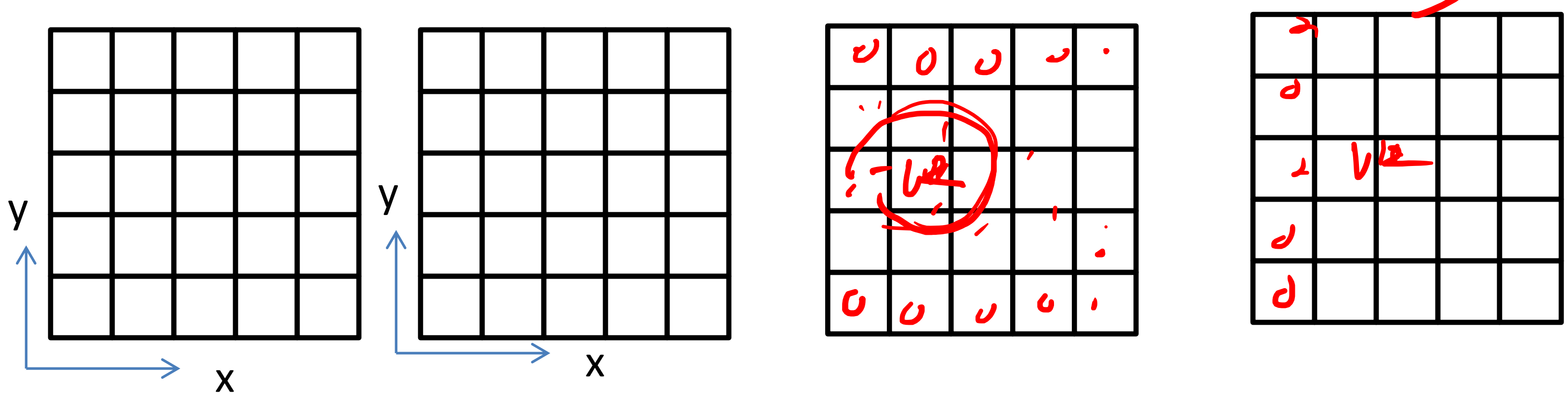
I_1, I_2

- We know to compute f_x, f_y, f_t,
- Initialize u, v, to 0. Estimate u_{av}, $v_{av=0}$
- Then iterate approximate 'u' and 'v' using previous set of equations, s.t,

$K=0$

$$u^k = u_{av}^{k-1} - f_x \frac{P}{D},$$
$$v^k = v_{av}^{k-1} - f_y \frac{P}{D}.$$

- error is minimized.



I(x,y,t) I(x,y,t+dt) 'u' 'v'

$$P = f_x u_{av} + f_y v_{av} + f_t, \text{ and } D = \lambda + {f_x}^2 + {f_y}^2$$

Horn and Schunck Optical Flow

global technique. of OF.

- The iterative algorithm can be given as

1. $k = 0$.
2. Initialize u^k and v^k to zero.
3. Until some error measure is satisfied, do:

$E = \iint E_d + \lambda E_s$

$$u^k = u_{av}^{k-1} - f_x \frac{P}{D},$$

$$v^k = v_{av}^{k-1} - f_y \frac{P}{D}.$$

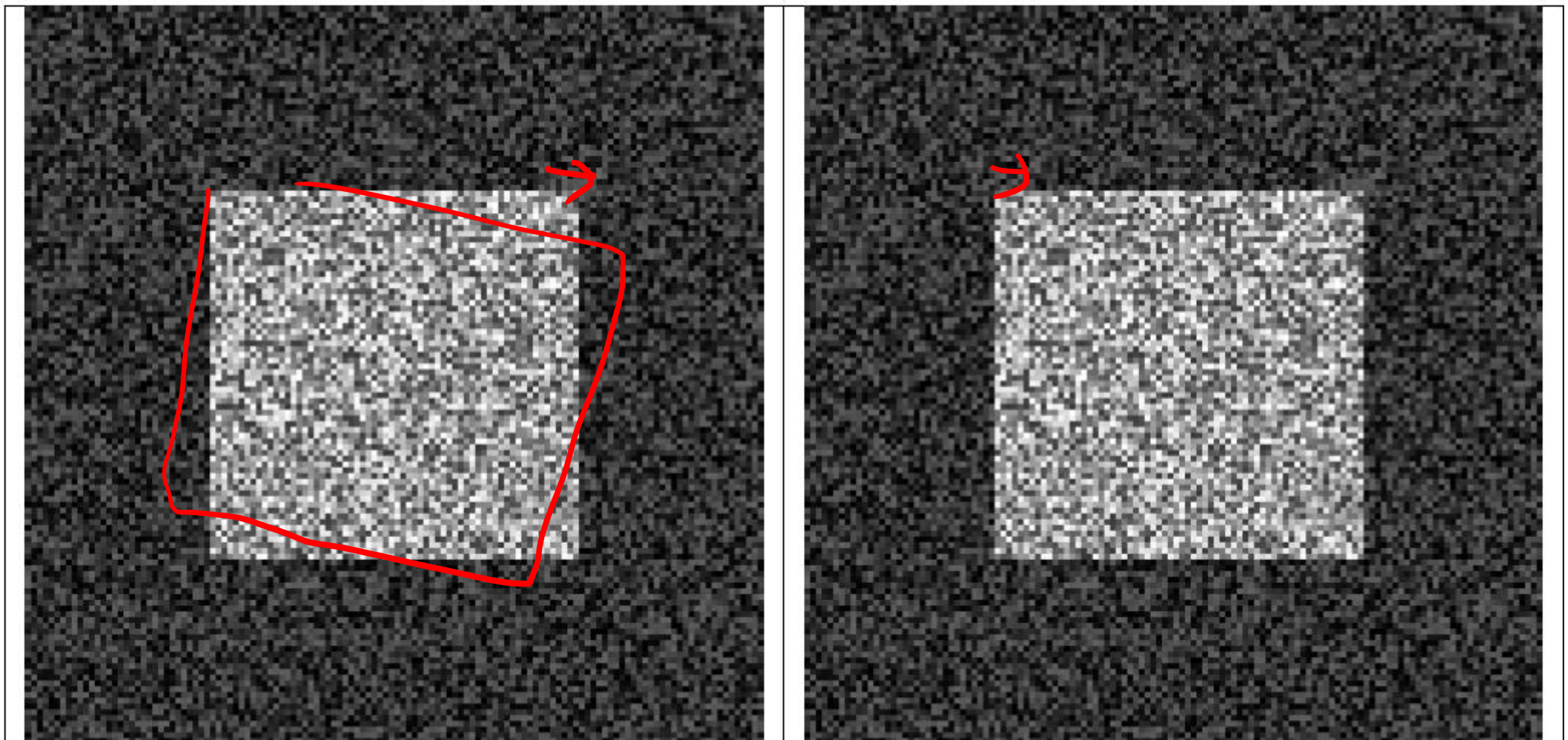
$P = f_x u_{av} + f_y v_{av} + f_t$, and $D = \lambda + {f_x}^2 + {f_y}^2$

- Using these u, v try to minimize

$$E_{\text{HS}}(u,v) = \sum_{x,y}\left(\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t}\right)^2 + \lambda\left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right)$$

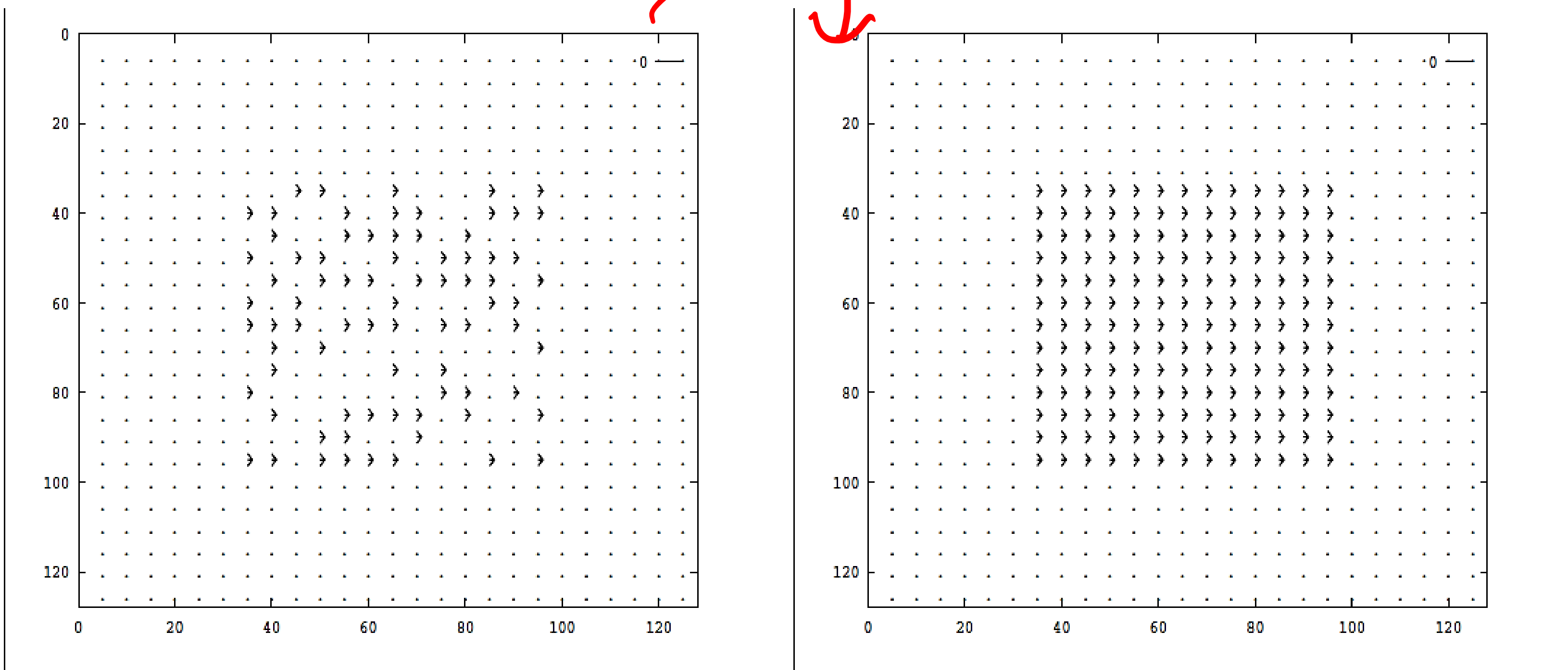
$$= E_{\text{data}}(u,v) + \lambda E_{\text{smoothness}}(u,v)$$

Horn and Schunck Optical Flow Example



(a)Frame 1 (b) Frame 2

Horn and Schunck Optical Flow Example

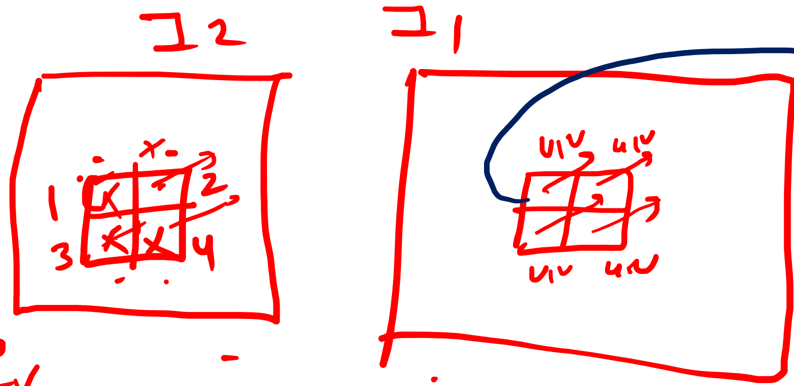


(c) 1 iteration (d) 10 iterations

Figure 5.4: Results for Horn and Schunck algorithm for displacement of 1 pixel and $\lambda = 4$.

ebook section of classroom

Book: FUNDAMENTALS OF **COMPUTER VISION** (**Mubarak Shah)**

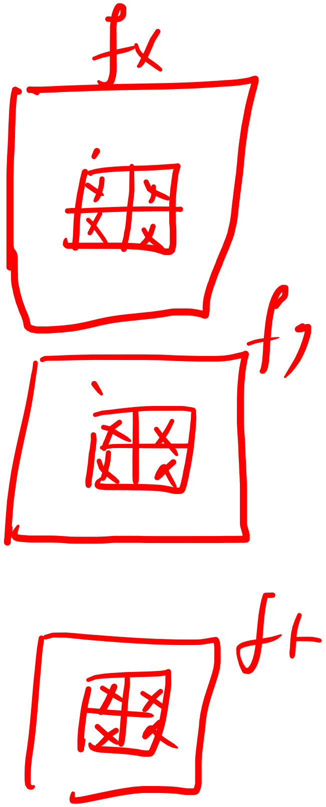


px1 $\boxed{f_{x_1} u + f_{y_1} v + f_t = 0}$

px2 $f_{x_2} u + f_{y_2} v + f_{t_1} = 0$

px3 $f_{x_3} u + f_{y_3} v + f_{t_2} = 0$

px4 $f_{x_4} u + f_{y_4} v + f_{t_3} = 0$



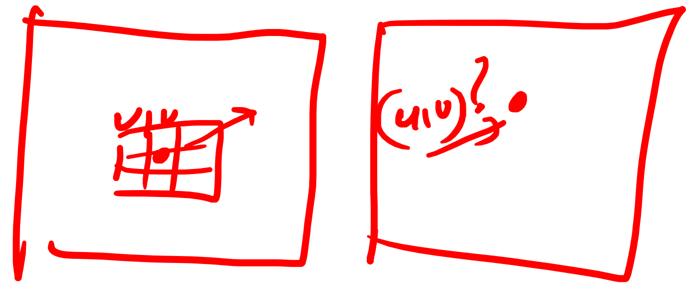
2. Lucas Kanade Approach

Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674.

Lucas & Kanade Method

- H&S is proposed a error minimizer using global approach
- L&K-> approach is local
- The optical flow for pixels belonging to a local neighborhood tends to be same.. Doesn’t changes drastically
- So, we work on a local window ‘w(x,y)’ to make error estimation
- Window size can be 3x3, 4x4,5x5……

Lucas & Kanade Method



- Optical flow eq

① $\boxed{f_x u + f_y v = -f_t}$ — BC.

- Consider 3 by 3 window

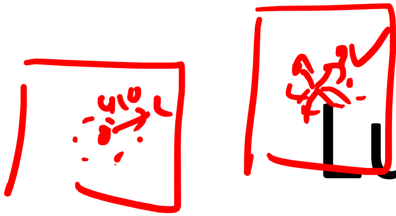
9 pixel

$$\left\{\begin{array}{l} f_{x1}u + f_{y1}v = -f_{t1} \\ \vdots \\ f_{x9}u + f_{y9}v = -f_{t9} \end{array}\right.$$

$$\text{find } (u,v) \text{ s.t. } \min \sum_{i=1}^{9} (f_{xi}u + f_{yi}v + f_t)^2$$

minimize 1 objective fn over a window w.r.t. (u,v)

Least Squares Fit



Lucas & Kanade Method

Estimate u, v!

Objective = over a window

- For all pixels in window $\min \sum_{i=1}^{w \times w} (f_{xi}u + f_{yi}v + f_t)^2$
- Differentiate wrt u,v

$$\frac{\partial}{\partial u}\sum_i (f_{xi}u + f_{yi}v + f_t)^2 = 0 \quad \rightarrow \quad \boxed{\sum (f_{xi}u + f_{yi}v + f_{ti})f_{xi} = 0} \quad (1)$$

$$\frac{\partial}{\partial v}\sum_i (f_{xi}u + f_{yi}v + f_t)^2 = 0 \qquad \boxed{\sum (f_{xi}u + f_{yi}v + f_{ti})f_{yi} = 0} \quad (2)$$

$$f_x u + f_y v + f_t = 0$$

Lucas & Kanade Method

- For all pixels in window

$\frac{\partial}{\partial u}$

$$\sum(f_{xi}u + f_{yi}v + f_{ti})f_{xi} = 0 \rightarrow 1$$

$\frac{\partial}{\partial v}$

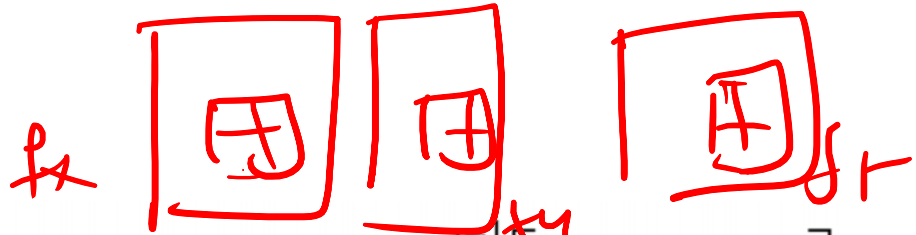
$$\sum(f_{xi}u + f_{yi}v + f_{ti})f_{yi} = 0 \rightarrow 2$$

=>
$$\sum f_{xi}^2 u + \sum f_{xi} f_{yi} v = -\sum f_{xi} f_{ti} \rightarrow 3$$
$$\sum f_{xi} f_{yi} u + \sum f_{yi}^2 v = -\sum f_{yi} f_{ti} \rightarrow 4$$

=>
$$\boxed{\underbrace{\begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{yi}^2 \end{bmatrix}}_{2\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{2\times 1} = \underbrace{\begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}}_{2\times 1} \rightarrow 5$$

inv

Lucas & Kanade Method



$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi} f_{yi} \\ \sum f_{xi} f_{yi} & \sum f_{yi}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2} \begin{bmatrix} \sum f_{yi}^2 & -\sum f_{xi} f_{yi} \\ -\sum f_{xi} f_{yi} & \sum f_{xi}^2 \end{bmatrix} \begin{bmatrix} -\sum f_{xi} f_{ti} \\ -\sum f_{yi} f_{ti} \end{bmatrix}$$

Lucas & Kanade Method

$$u = \frac{-\sum f_{yi}^2 \sum f_{xi} f_{ti} + \sum f_{xi} f_{yi} \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2}$$

$$v = \frac{\sum f_{xi} f_{ti} \sum f_{xi} f_{yi} - \sum f_{xi}^2 \sum f_{yi} f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi} f_{yi})^2}$$

Lucas-Kanade Variations

- We now want to minimize error given by

$$E_{\mathrm{LK}}(u,v) = \sum_{(x,y)} w(x,y)\left(\overset{I_1}{I(x+u,y+v,t+1)} - \overset{I_2}{I(x,y,t)}\right)^2$$

- where $w(x,y)$ is a window function centered at (x_0, y_0) (for example, a box filter, or a Gaussian with a given scale σ).
- Effectively, we're assuming that all the pixels in the window have the same motion vector.
- Using large windows may not do a very good job of estimating flow.

Lucas-Kanade Method Variations

- The minimizer corresponds to the solution of the linear system

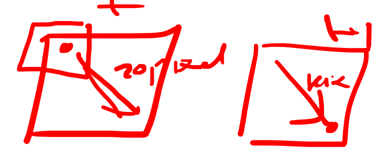
$$\begin{bmatrix} \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial x}(x,y)\right)^2 & \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y)\right) \\ \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial y}(x,y)\right) & \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial y}(x,y)\right)^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$= -\begin{bmatrix} \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial x}(x,y)\frac{\partial I}{\partial t}(x,y)\right) \\ \sum_{(x,y)} w(x,y)\left(\frac{\partial I}{\partial y}(x,y)\frac{\partial I}{\partial t}(x,y)\right) \end{bmatrix}$$

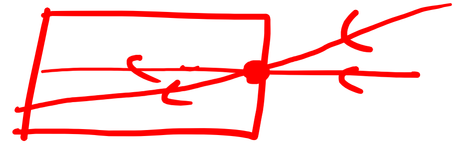
Computer Vision for Visual Effects

Richard J. Radke, Cambridge University Press, 2012 (Page no: 160)

H&S
L&K



Drawbacks Optical Flow



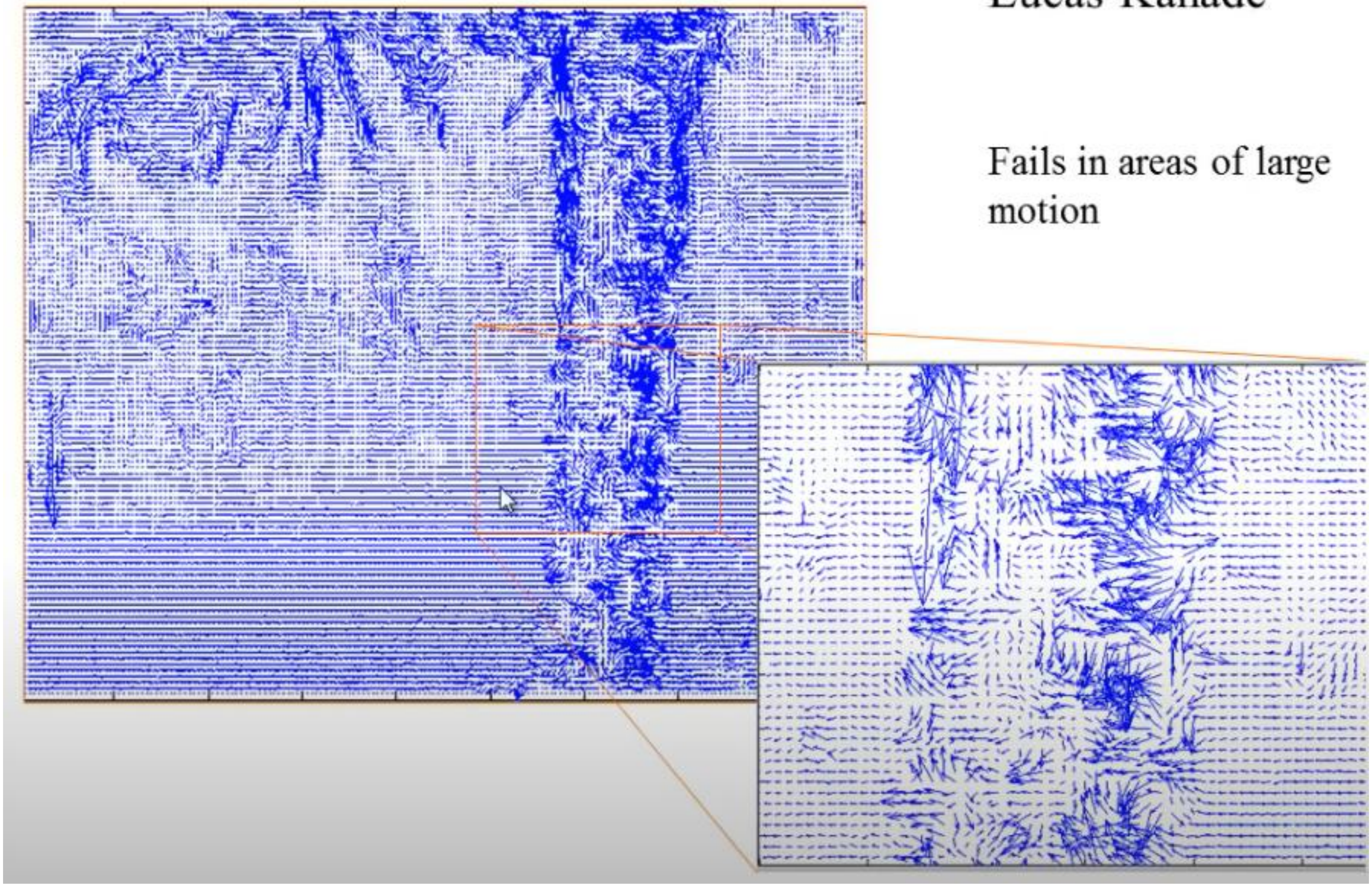
Drawbacks

- H&S: Taylor series approximation is only valid when *(u,v)* is close to zero.

$$f(x, y, t) - f(x + dx, y + dy, t + dt) = 0$$

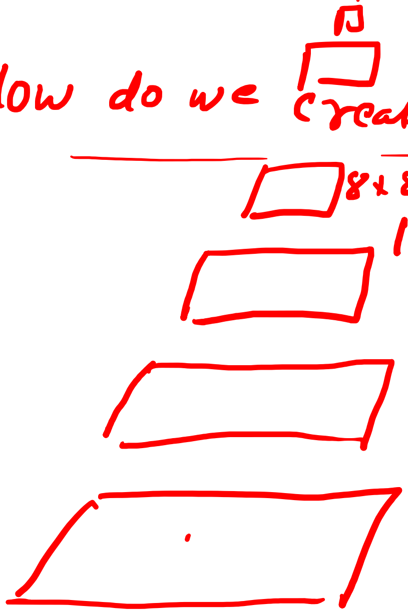
- Captures only small motion , that is, with *u* and *v* each less than one pixel.
- If object moves faster, the brightness changes rapidly, – 2x2 or 3x3 masks fail to estimate spatiotemporal derivatives.
- This issue can be addressed using a **hierarchical** or **multiresolution** approach

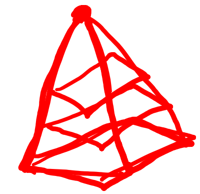
Lucas-Kanade



Fails in areas of large motion

How do we create Image pyramid





8x8

16x16

32x32

smooth using GF & desample to reduce size by half

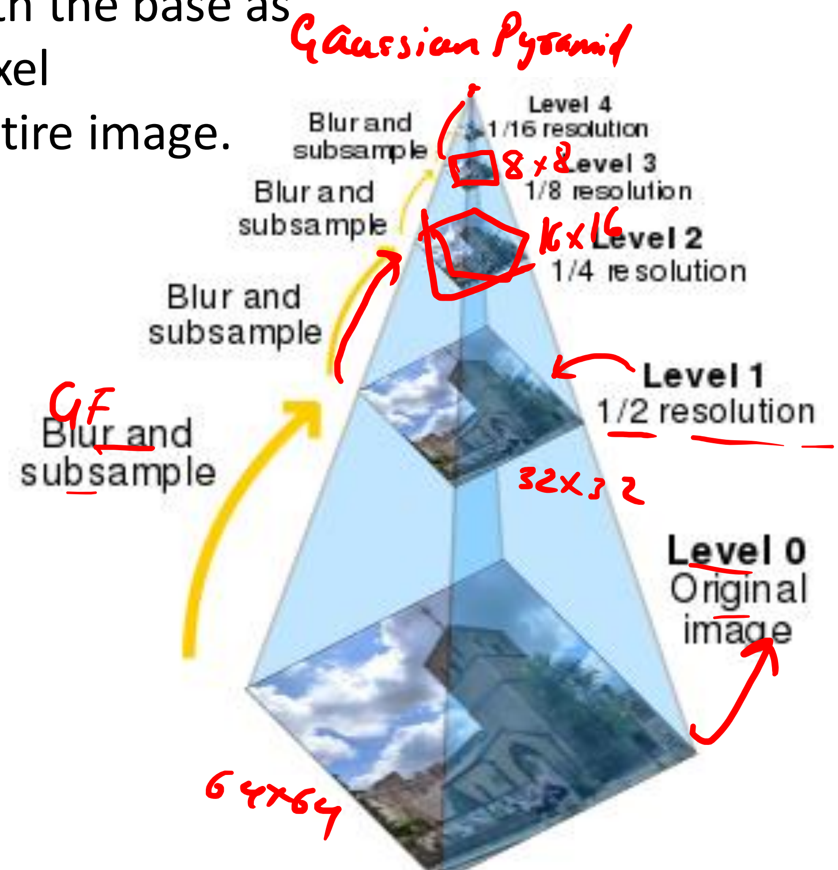
64x64

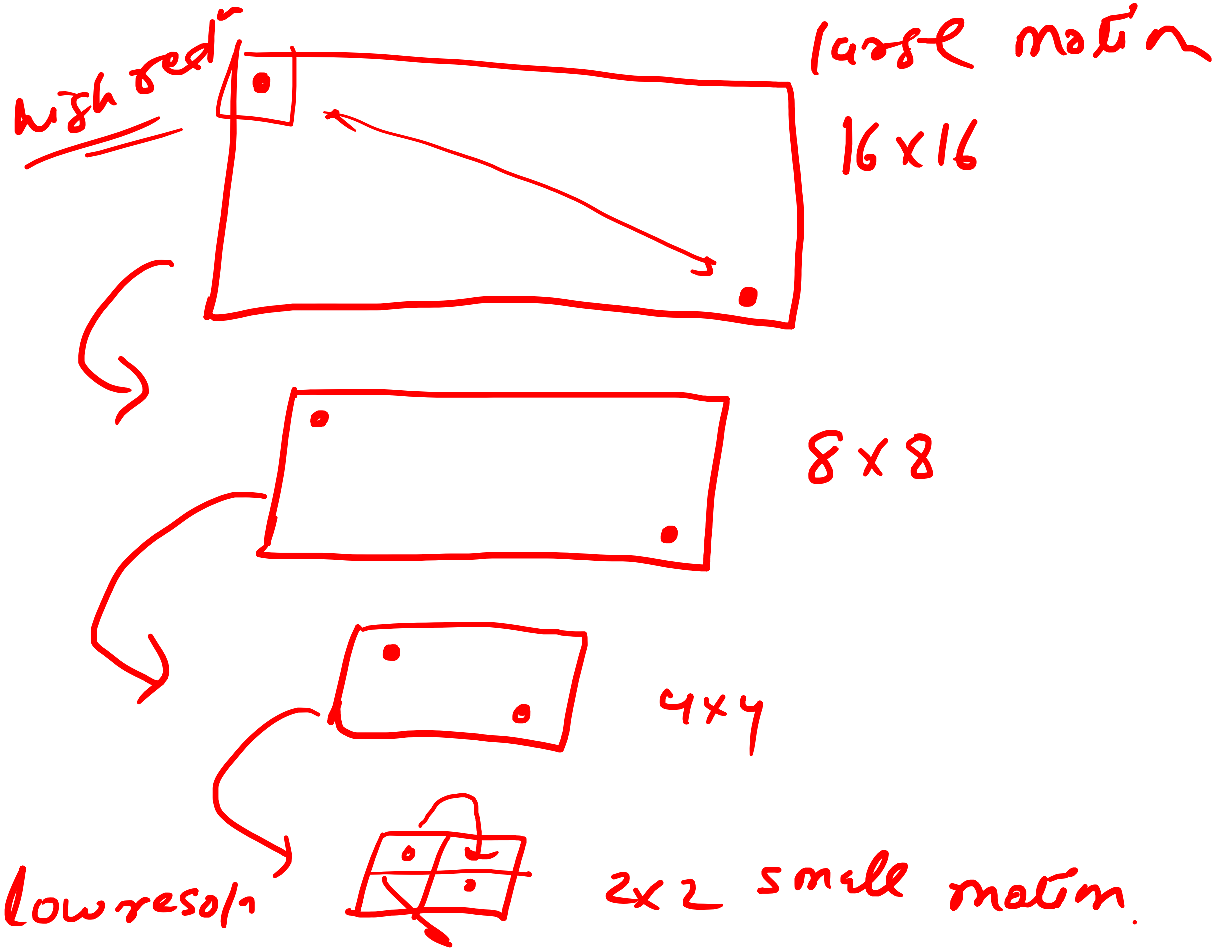
3. Pyramid Method

J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision (ECCV)*, 1992.

Pyramids

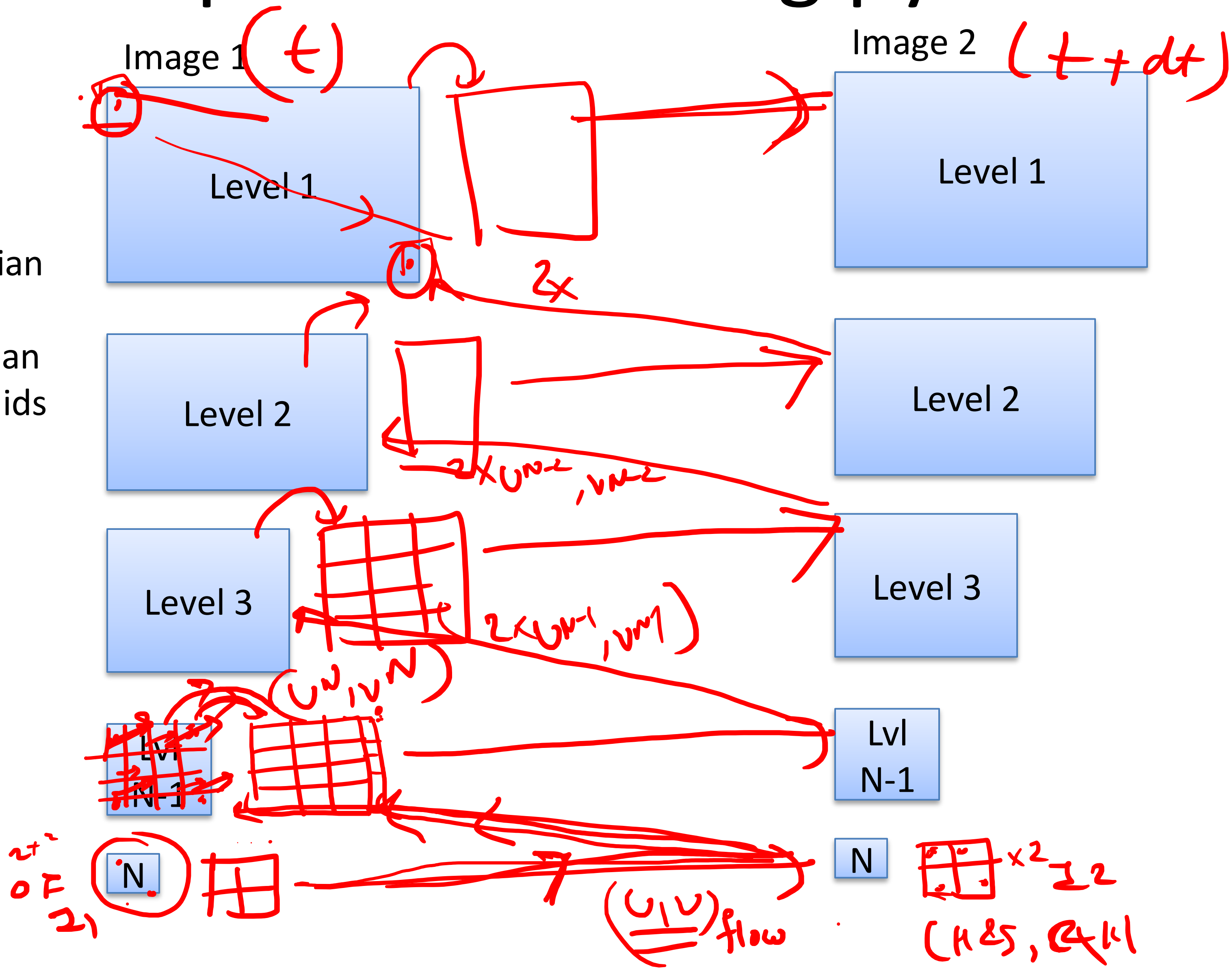
- The "pyramid" is constructed by repeatedly calculating a weighted average of the neighboring pixels of a source image and scaling the image down.
- This process creates a pyramid shape with the base as the original image and the tip a single pixel representing the average value of the entire image.



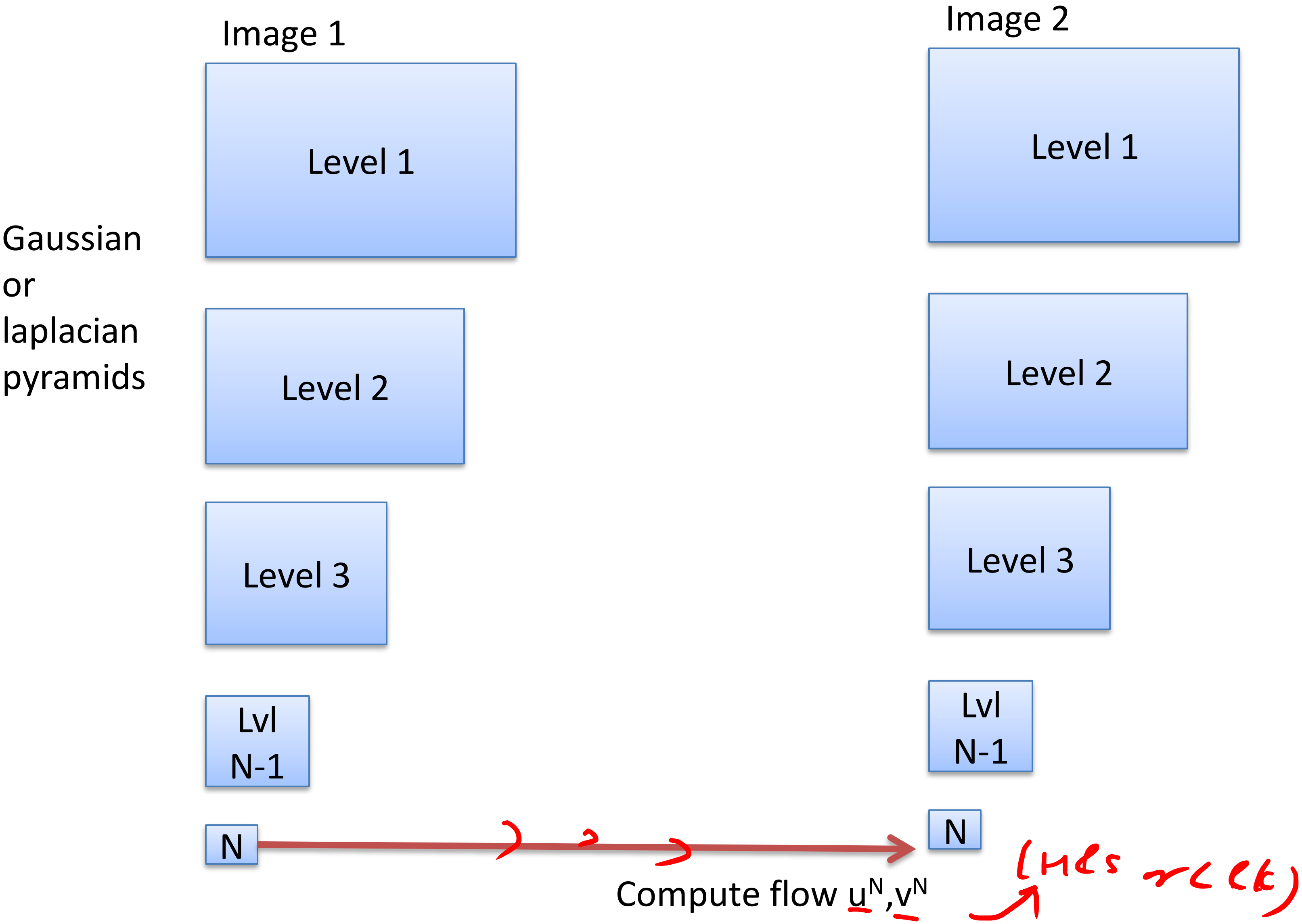

high resol^n
large matrix
16x16
8x8
4x4
low resol^n
2x2 small matrix

Optical flow using pyramids

Gaussian or laplacian pyramids

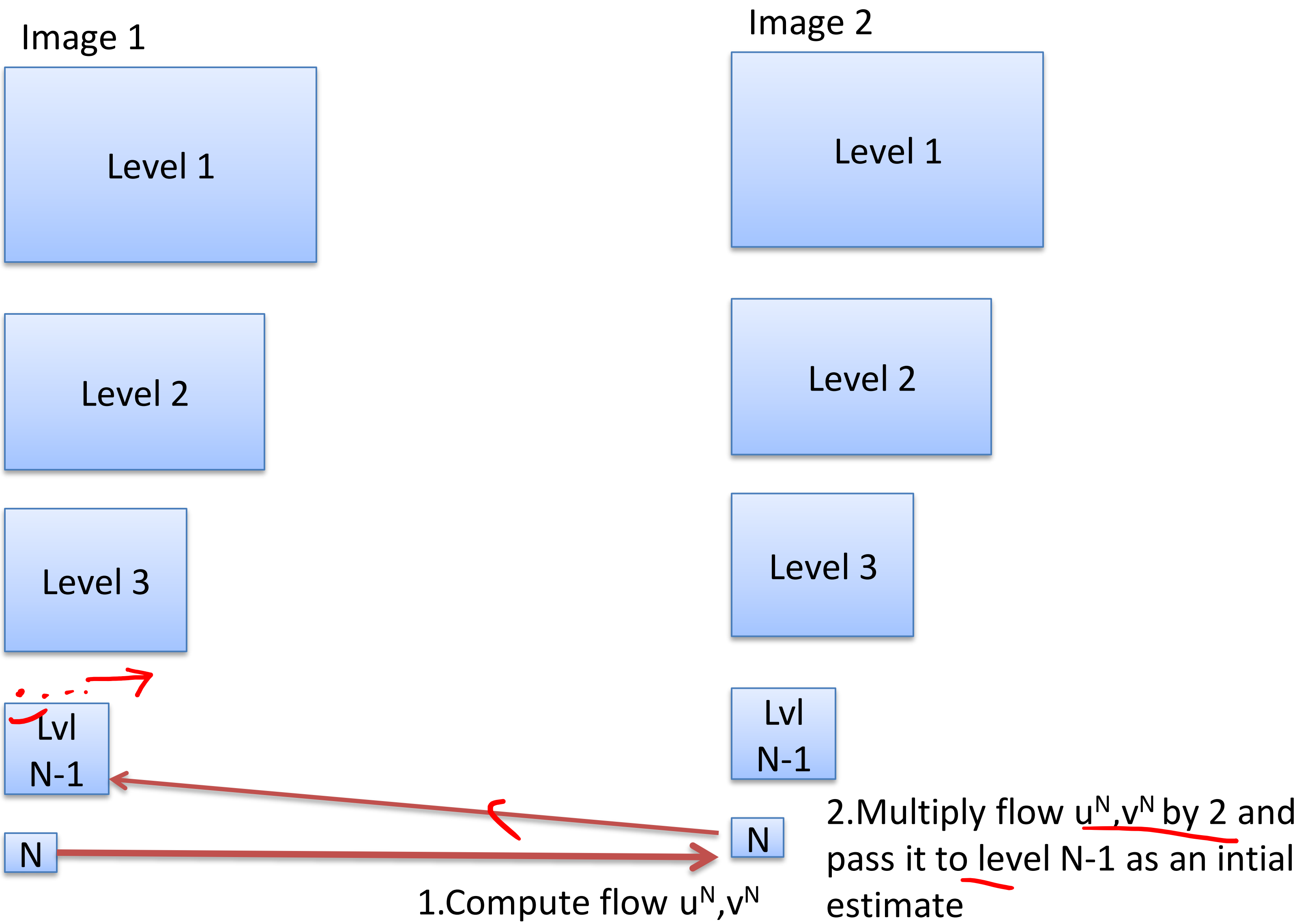


Optical flow using pyramids



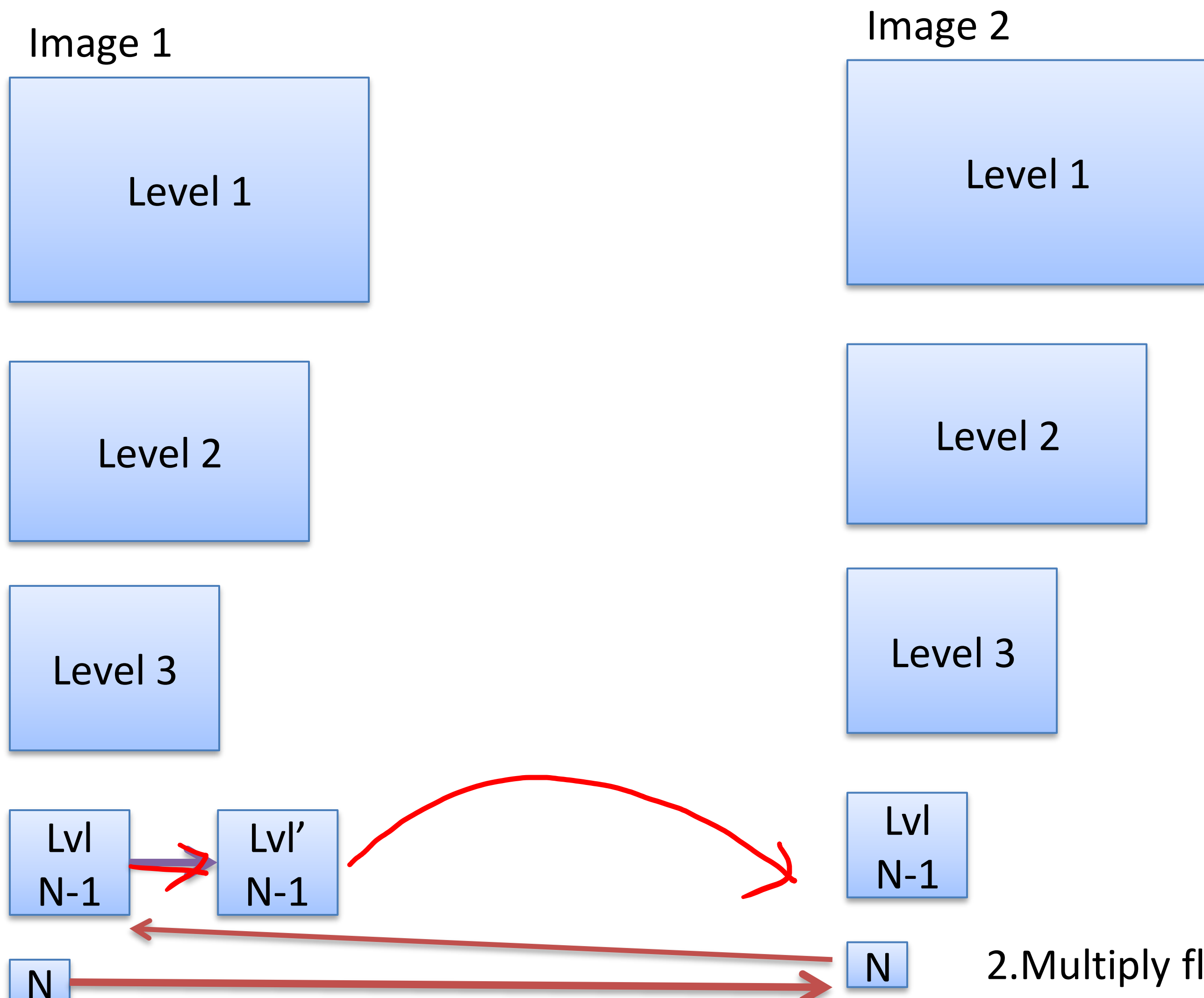
Optical flow using pyramids

Gaussian or laplacian pyramids



Optical flow using pyramids

Gaussian or laplacian pyramids

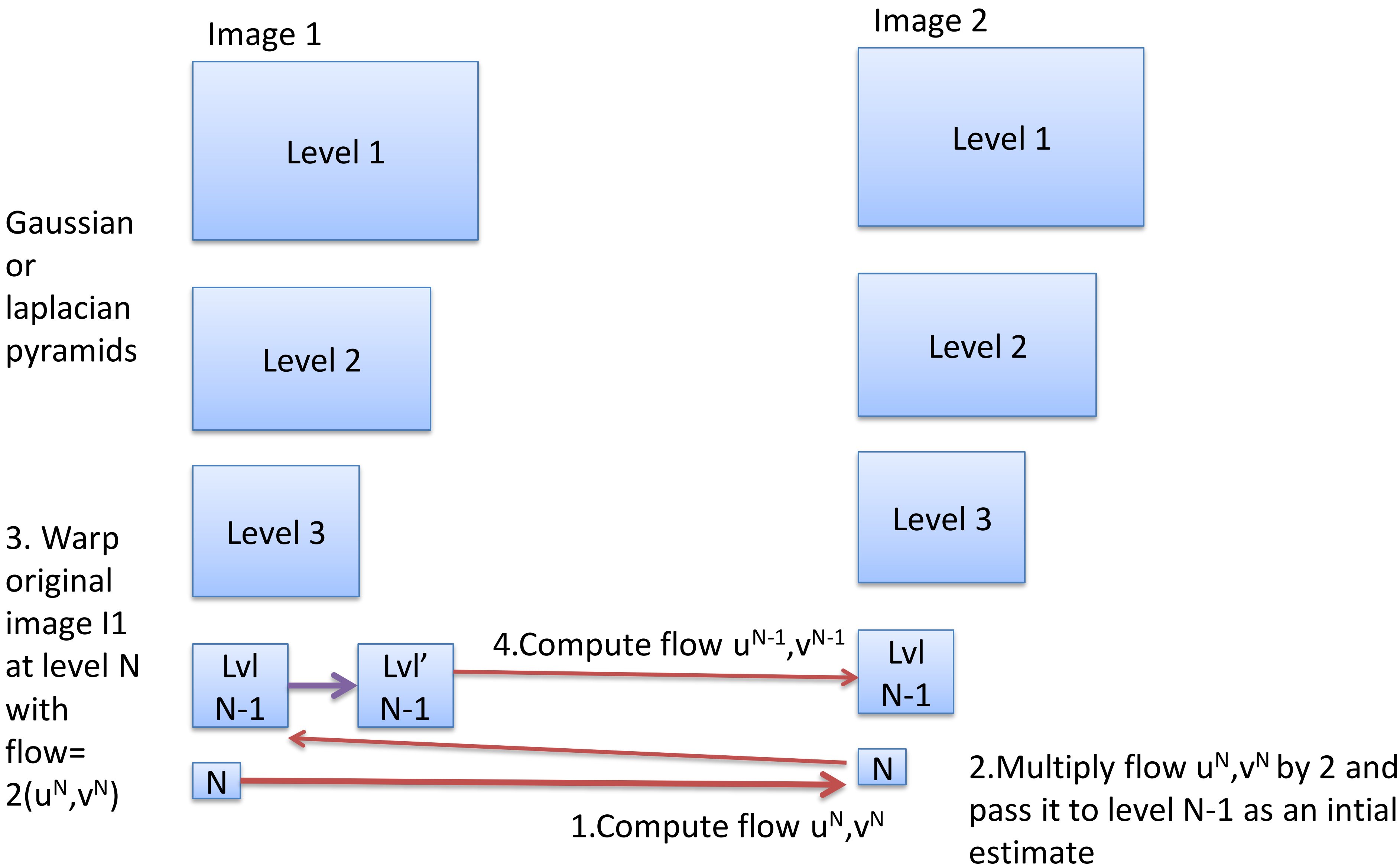


3. Warp original image I1 at level N-1 with flow= $2(u^N,v^N)$

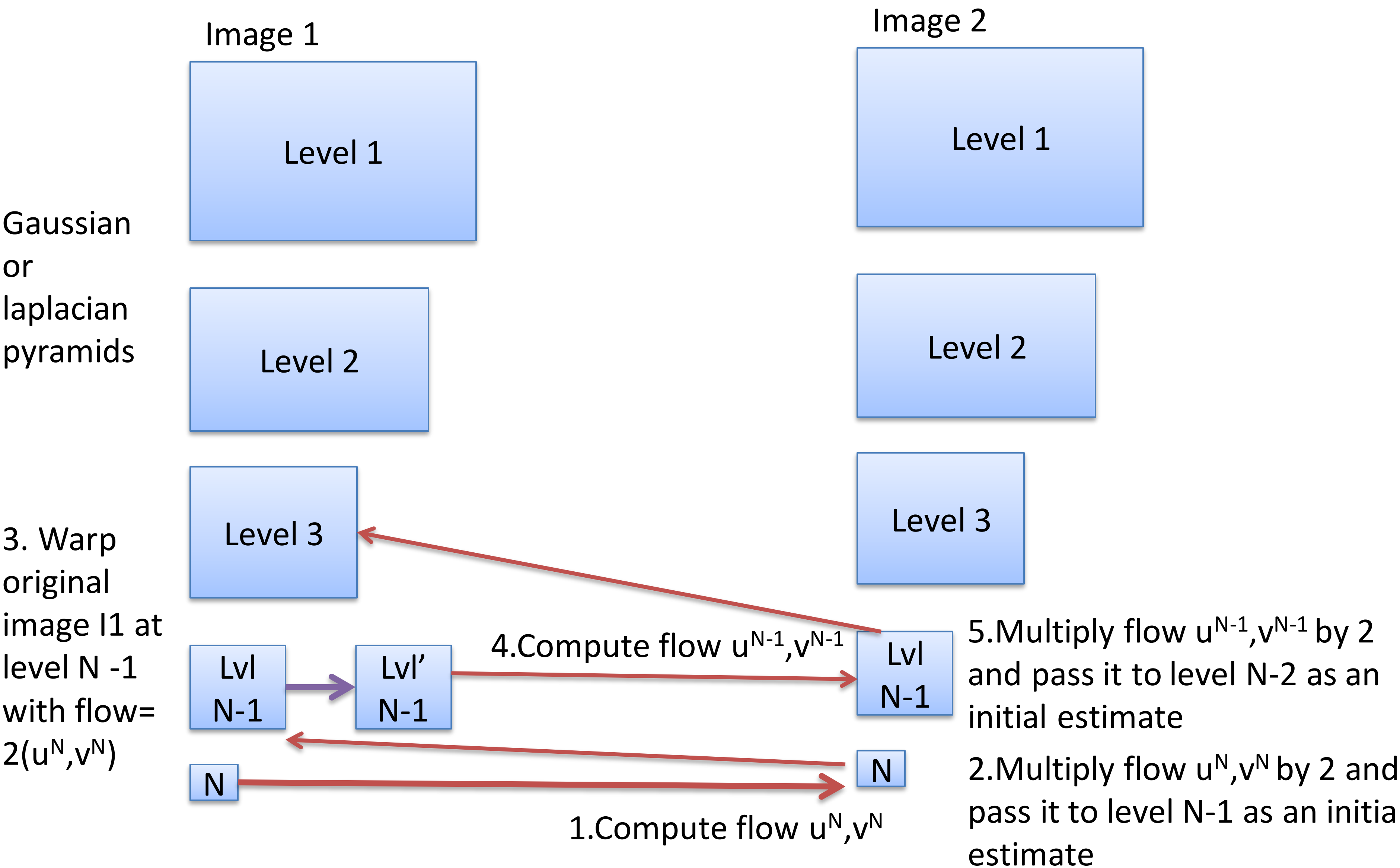
1.Compute flow u^N,v^N

2.Multiply flow u^N,v^N by 2 and pass it to level N-1 as an intial estimate

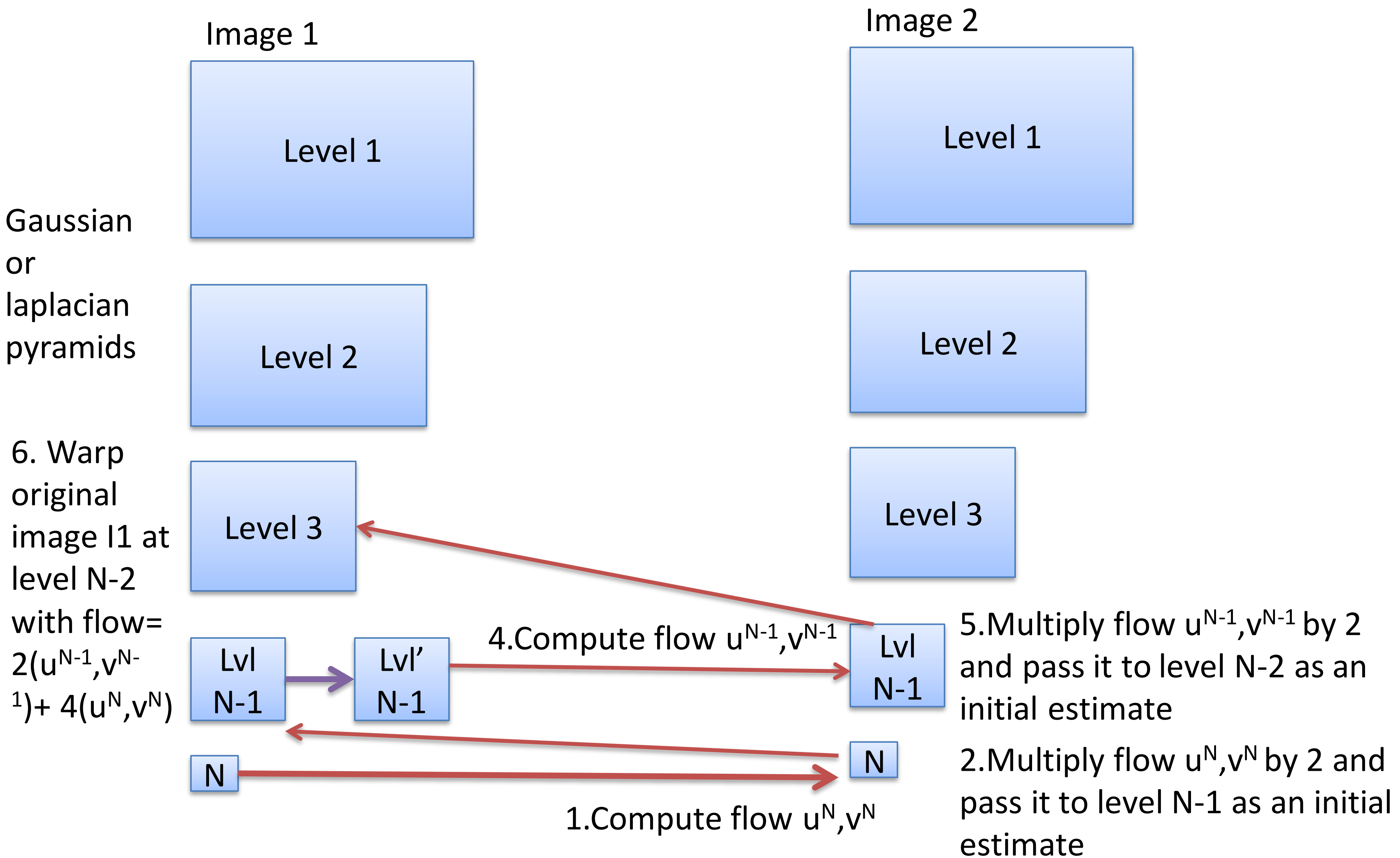
Optical flow using pyramids



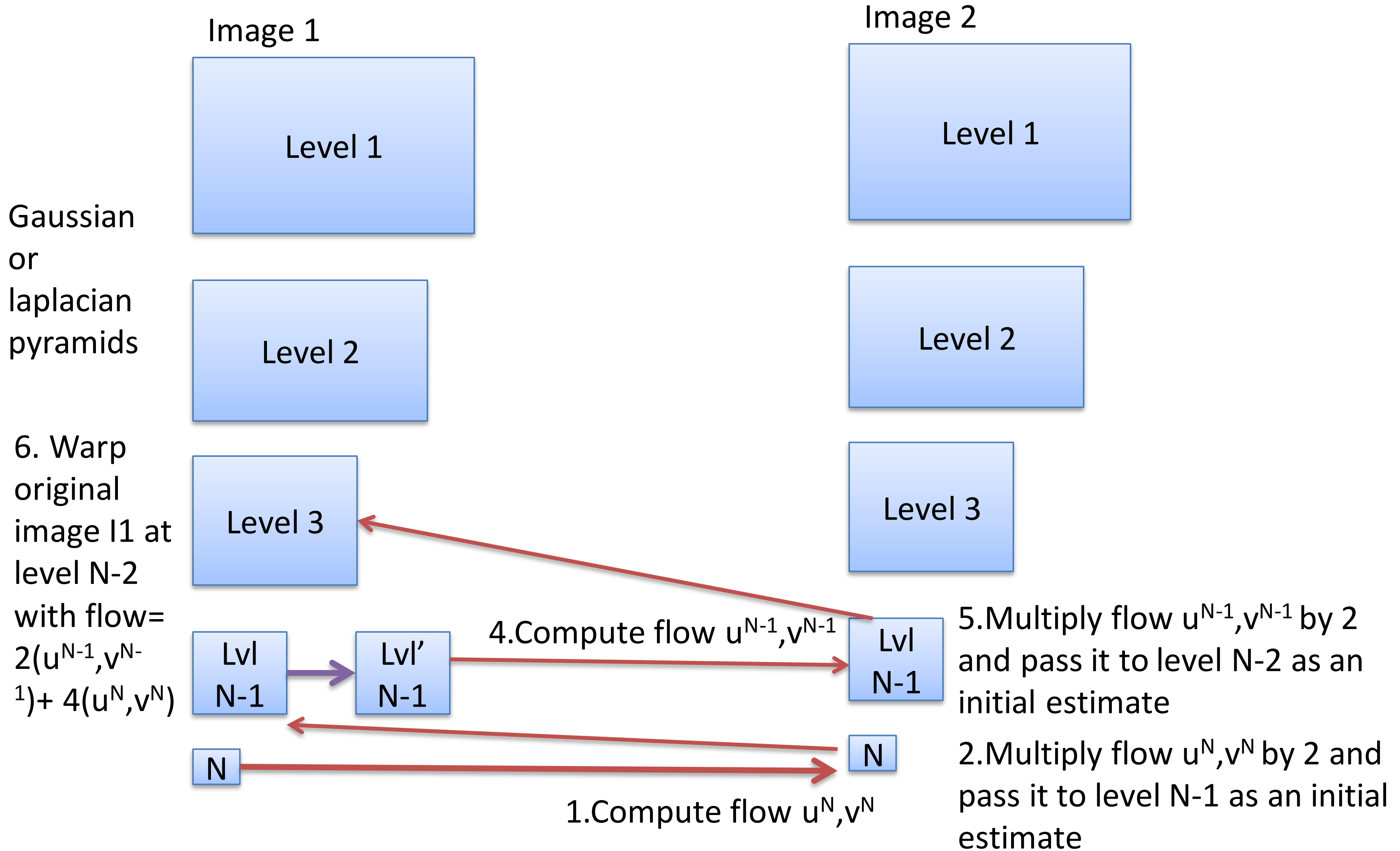
Optical flow using pyramids



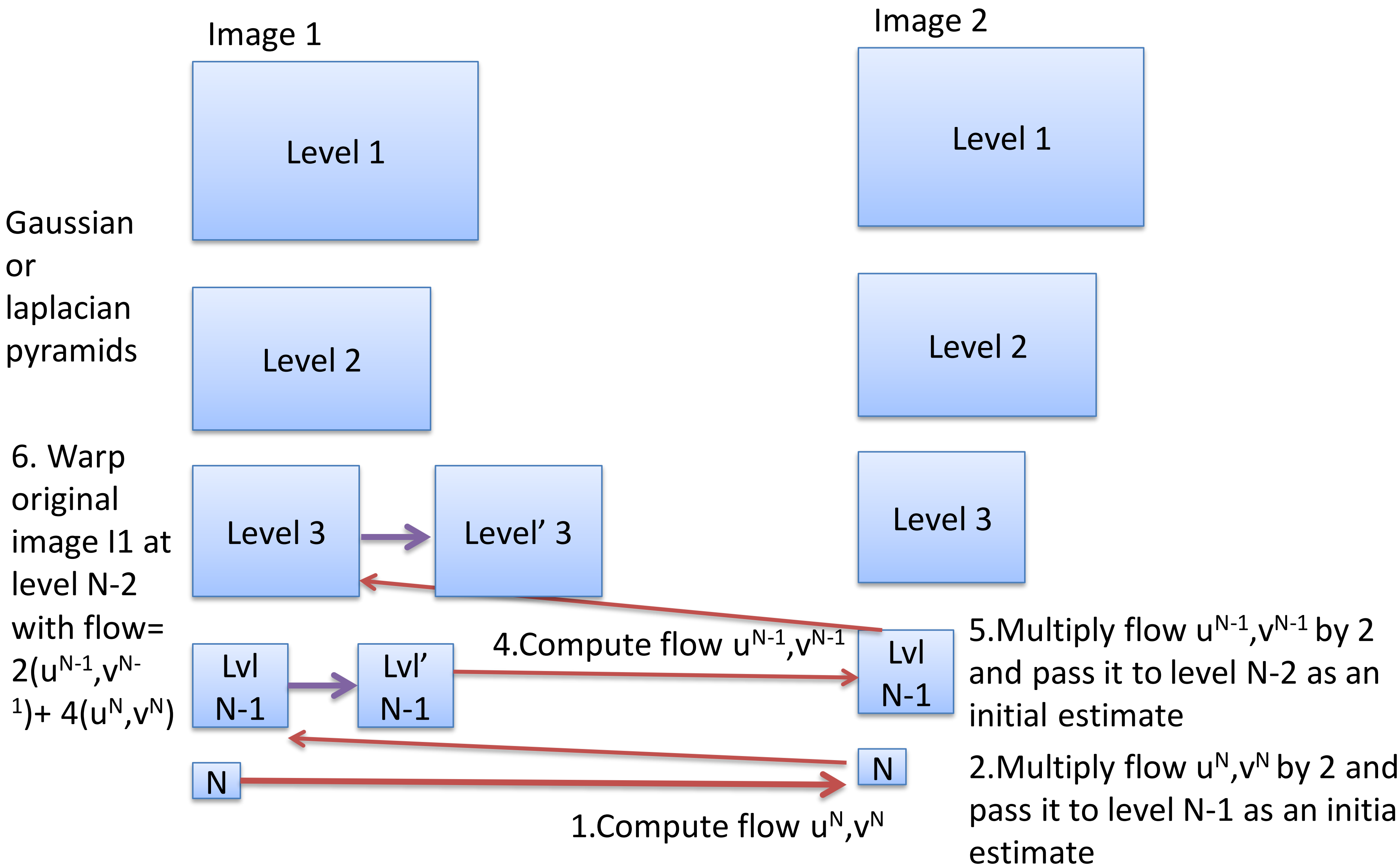
Optical flow using pyramids



Optical flow using pyramids



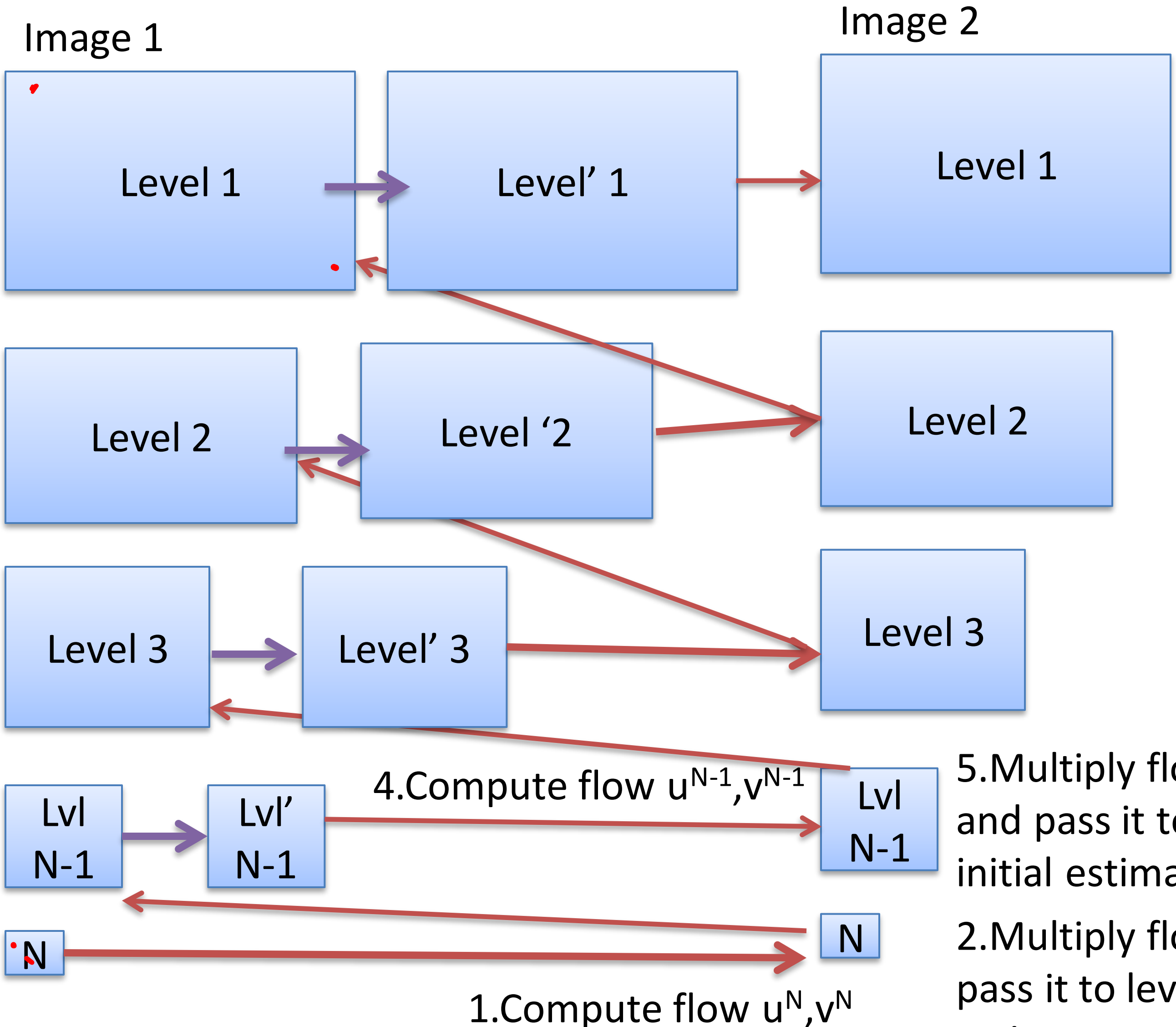
Optical flow using pyramids



Optical flow using pyramids

Gaussian or laplacian pyramids

6. Warp original image I1 at level N-2 with flow= $2(u^{N-1},v^{N-1})+ 4(u^N,v^N)$



5.Multiply flow u^{N-1},v^{N-1} by 2 and pass it to level N-2 as an initial estimate

2.Multiply flow u^N,v^N by 2 and pass it to level N-1 as an initial estimate

Optical flow using pyramids: Algo

1. Create the Laplacian pyramids for I_1 and I_2, indexed from 0 to N, where N is the coarsest level of the pyramid. Initialize the working level n to N and let $\tilde{I}_1$ and $\tilde{I}_2$ be the images at the coarsest level of the pyramid, i.e., $\tilde{I}_1 = I_1^N$, $\tilde{I}_2 = I_2^N$.
2. Estimate the optical flow field $(u, v)^{(n)}$ between $\tilde{I}_1$ and $\tilde{I}_2$ at level n.

[Computer Vision for Visual Effects]()

Richard J. Radke, Cambridge University Press, 2012 (Page no: 159)

Optical flow using pyramids: Algo

1. Create the Laplacian pyramids for I_1 and I_2, indexed from 0 to N, where N is the coarsest level of the pyramid. Initialize the working level n to N and let $\tilde{I}_1$ and $\tilde{I}_2$ be the images at the coarsest level of the pyramid, i.e., $\tilde{I}_1 = I_1^N$, $\tilde{I}_2 = I_2^N$.
2. Estimate the optical flow field $(u,v)^{(n)}$ between $\tilde{I}_1$ and $\tilde{I}_2$ at level n.
3. Construct the optical flow field $(u,v)^{\text{warp}}$ as $\sum_{k=n}^{N} 2^{k-n+1}(u,v)^{(k)}$ — that is, the sum of all the incremental motion fields so far, at the scale of level $n-1$. Since this generates motion vector estimates only for even x and y, use bilinear interpolation to fill in motion vectors for the whole image.
4. Warp the pyramid image for I_2 at level $n-1$ using $(u,v)^{\text{warp}}$ to create a new image $\tilde{I}_2$. That is, $\tilde{I}_2(x,y) = I_2^{n-1}(x+u^{\text{warp}}, y+v^{\text{warp}})$. Use bilinear interpolation to sample pixels from I_2^{n-1}. Let $\tilde{I}_1 = I_1^{n-1}$.

[Computer Vision for Visual Effects]() ebook

Richard J. Radke, Cambridge University Press, 2012 (Page no: 159)

Optical flow using pyramids: Algo

1. Create the Laplacian pyramids for I_1 and I_2, indexed from 0 to N, where N is the coarsest level of the pyramid. Initialize the working level n to N and let $\tilde{I}_1$ and $\tilde{I}_2$ be the images at the coarsest level of the pyramid, i.e., $\tilde{I}_1 = I_1^N$, $\tilde{I}_2 = I_2^N$.
2. Estimate the optical flow field $(u, v)^{(n)}$ between $\tilde{I}_1$ and $\tilde{I}_2$ at level n.
3. Construct the optical flow field $(u, v)^{\text{warp}}$ as $\sum_{k=n}^{N} 2^{k-n+1}(u, v)^{(k)}$ — that is, the sum of all the incremental motion fields so far, at the scale of level $n-1$. Since this generates motion vector estimates only for even x and y, use bilinear interpolation to fill in motion vectors for the whole image.
4. Warp the pyramid image for I_2 at level $n-1$ using $(u, v)^{\text{warp}}$ to create a new image $\tilde{I}_2$. That is, $\tilde{I}_2(x, y) = I_2^{n-1}(x + u^{\text{warp}}, y + v^{\text{warp}})$. Use bilinear interpolation to sample pixels from I_2^{n-1}. Let $\tilde{I}_1 = I_1^{n-1}$.
5. Let $n = n - 1$.
6. If $n \geq 0$, go to Step 2. Otherwise, the final optical flow field is $(u, v) = \sum_{k=0}^{N} 2^k (u, v)^{(k)}$.

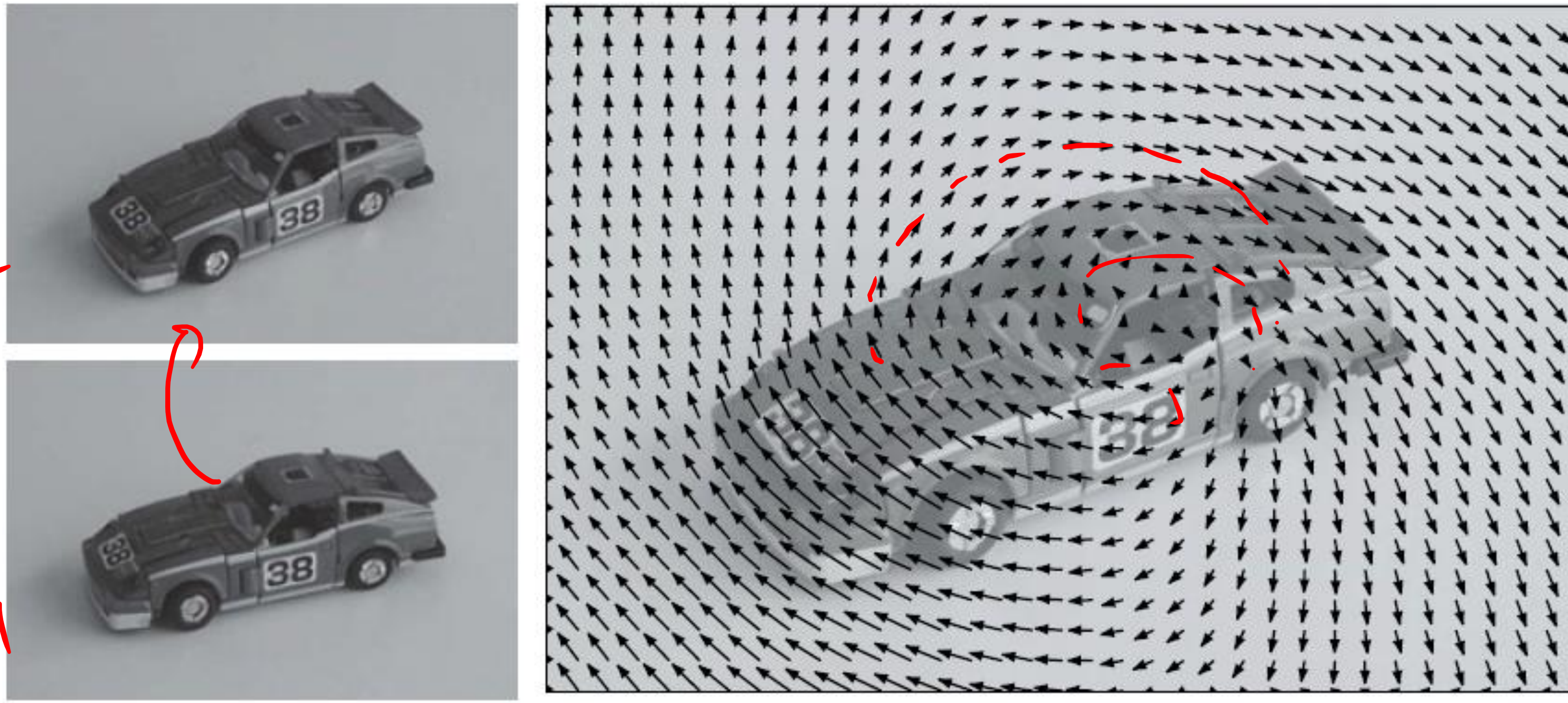
[Computer Vision for Visual Effects]

Richard J. Radke, Cambridge University Press, 2012 (Page no: 159)

Optical flow using pyramids: Ex

I_1

I_2



(a) (b)

Figure 5.5. Optical flow computed using a hierarchical implementation of the Horn Schunck algorithm. (a) Image 1 (top) and Image 2 (bottom). (b) Optical flow field overlaid on Image 1, indicated by arrows (only a sampling of arrows are illustrated for visibility). We can see that the flow field accurately captures the motion introduced by camera rotation, even in flat regions.

Refinements and Extensions

- The original Horn-Schunck data term encapsulates the assumption that pixel intensities don't change over time, which isn't realistic for many real-world scenes.
- Uras et al. proposed instead the **gradient constancy assumption**

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

- This allows some local variation to illumination changes

[Computer Vision for Visual Effects]
Richard J. Radke, Cambridge University Press, 2012 (Page no: 171)

Refinements and Extensions

- Bruhn et al. proposed to replace the Horn-Schunck ***data term*** with one inspired by the Lucas-Kanade algorithm, that is

$$E_{\text{data}}(u,v) = \sum_{(x,y)} w(x,y)(I_2(x+u,y+v) - I_1(x,y))^2$$

- This approach combines the advantages of Lucas-Kanade's local spatial smoothing, which makes the data term robust to noise, with Horn-Schunck's global regularization, which makes the flow fields smooth and dense.

[Computer Vision for Visual Effects]

Richard J. Radke, Cambridge University Press, 2012 (Page no: 172)