Computer Vision

CSE 455
Motion and Optical Flow

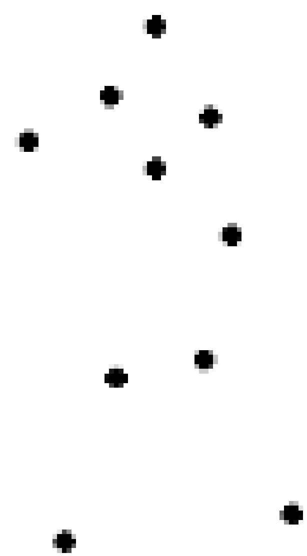
Linda Shapiro
Professor of Computer Science & Engineering
Professor of Electrical Engineering

We live in a moving world

- Perceiving, understanding and predicting motion is an important part of our daily lives

Motion and perceptual organization

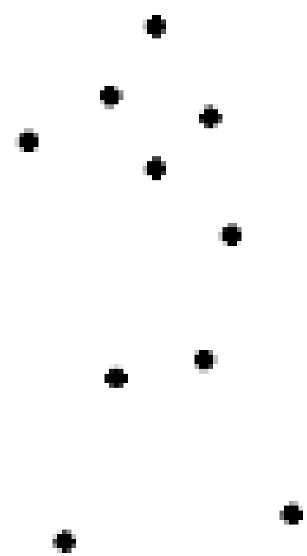
- Even “impoverished” motion data can evoke a strong percept



G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

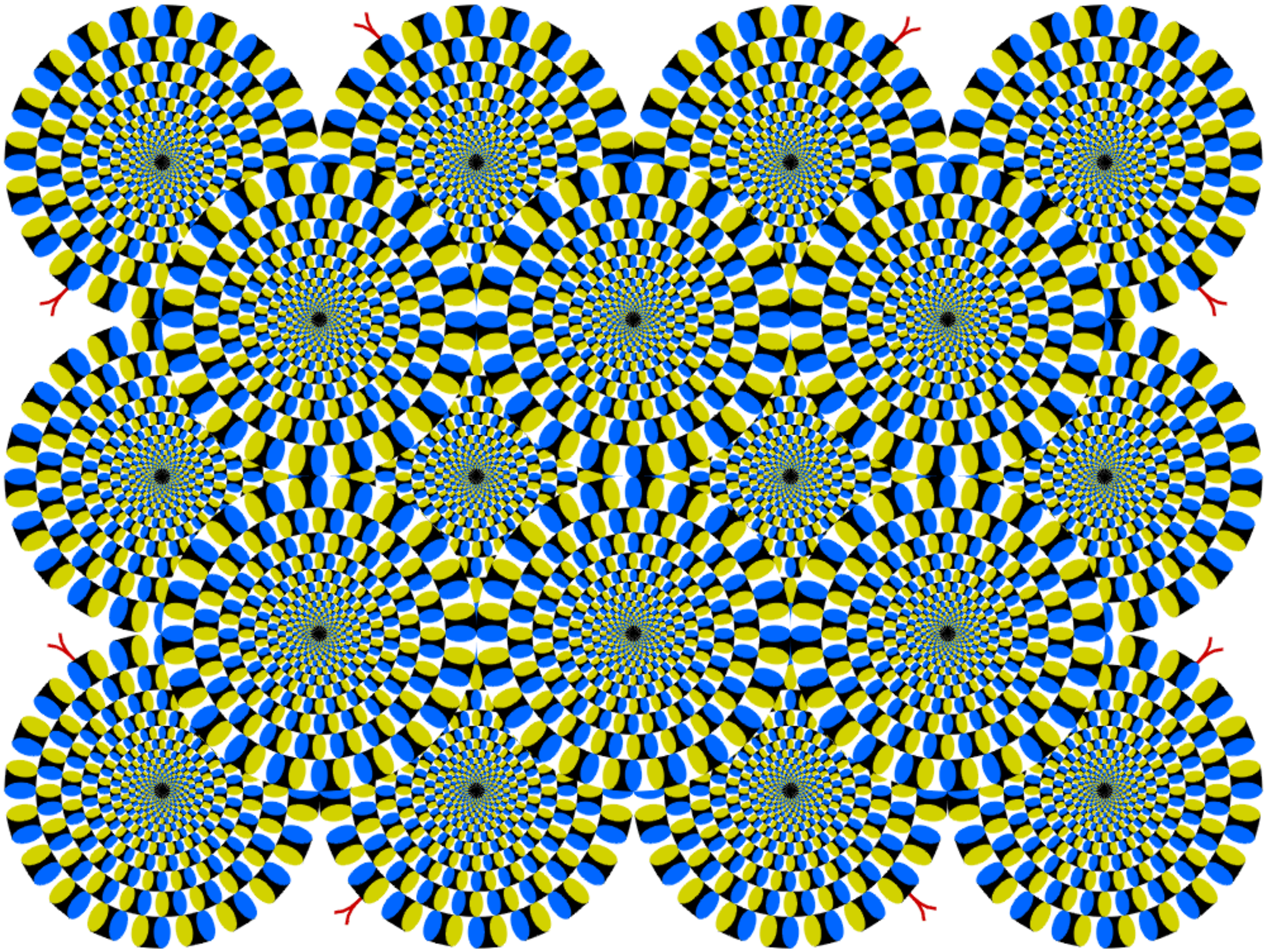
Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept



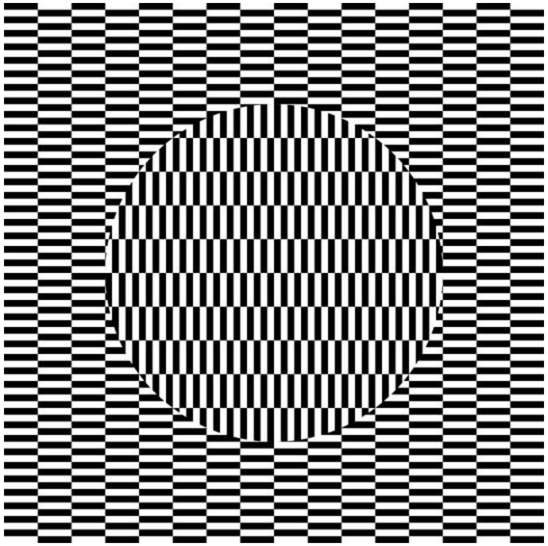
G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis", *Perception and Psychophysics 14, 201-211, 1973.*

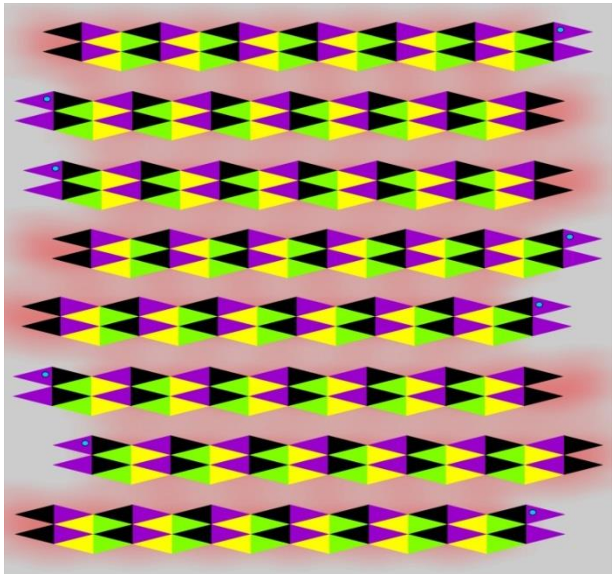
Seeing motion from a static picture?

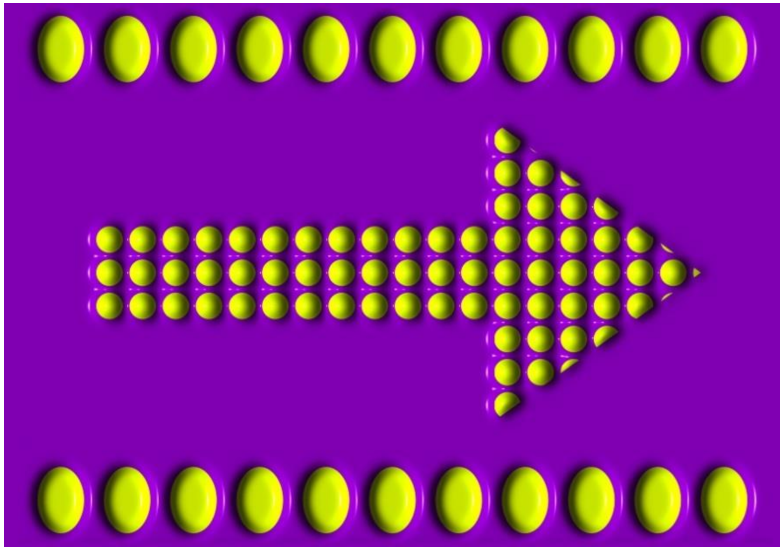


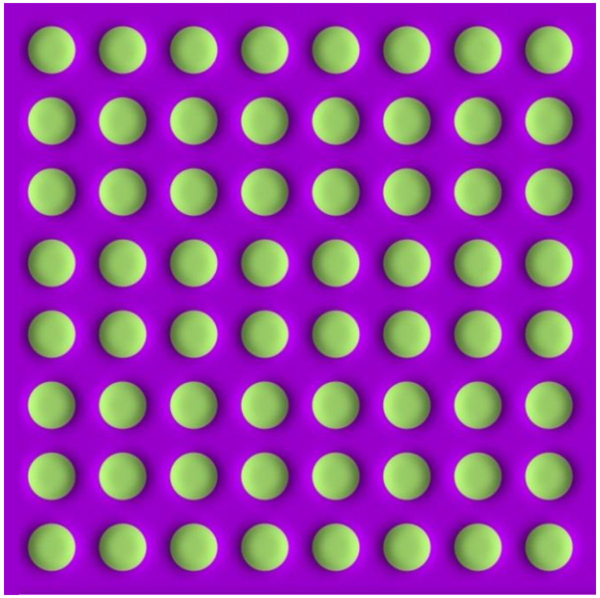
http://www.ritsumei.ac.jp/~akitaoka/index-e.html

More examples



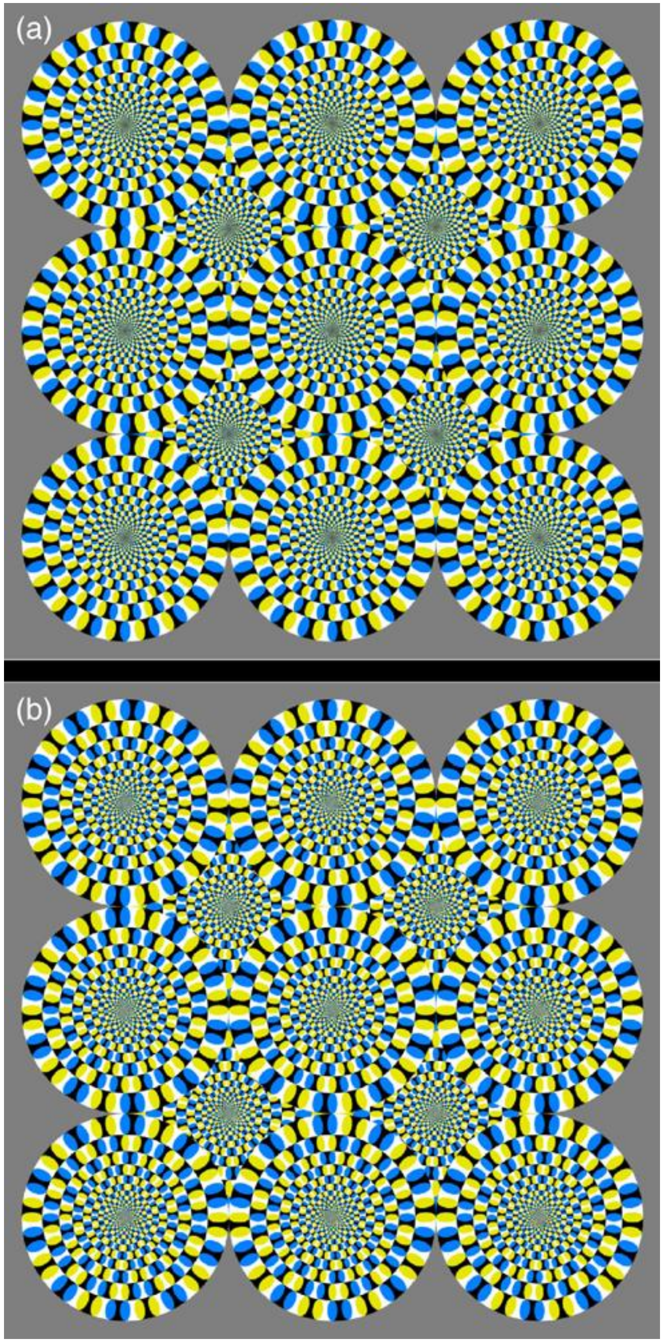






How is this possible?

- The true mechanism is yet to be revealed
- FMRI data suggest that illusion is related to some component of eye movements
- We don’t expect computer vision to “see” motion from these stimuli, yet



The cause of motion

- Three factors in imaging process
  - Light
  - Object
  - Camera

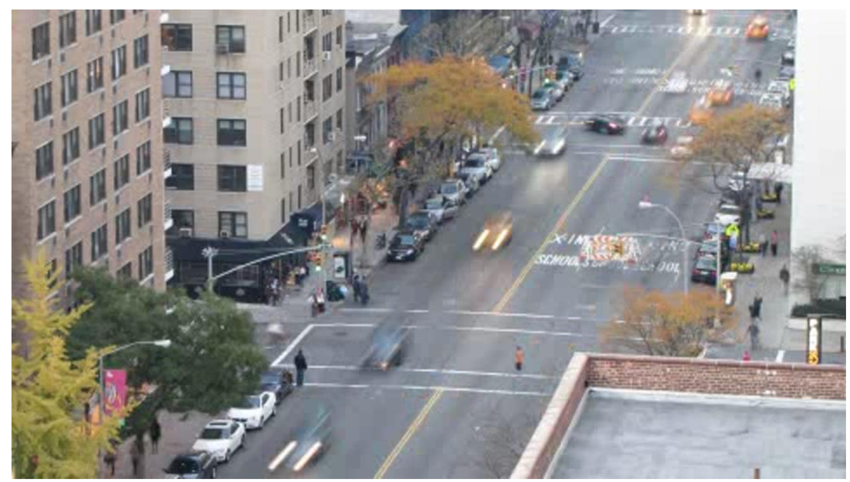
- Varying either of them causes motion
  - Static camera, moving objects (surveillance)
  - Moving camera, static scene (3D capture)
  - Moving camera, moving scene (sports, movie)
  - Static camera, moving objects, moving light (time lapse)

Motion scenarios (priors)

Static camera, moving scene

Moving camera, static scene

Moving camera, moving scene



Static camera, moving scene, moving light

We still don’t touch these areas

How can we recover motion?

Recovering motion

- Feature-tracking
  - Extract visual features (corners, textured areas) and "track" them over multiple frames

- Optical flow
  - Recover image motion at each pixel from spatio-temporal image brightness variations (optical flow)

Two problems, one registration method

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Feature tracking

- Challenges
  - Figure out which features can be tracked
  - Efficiently track across frames
  - Some points may change appearance over time (e.g., due to rotation, moving into shadows, etc.)
  - Drift: small errors can accumulate as appearance model is updated
  - Points may appear or disappear: need to be able to add/delete tracked points

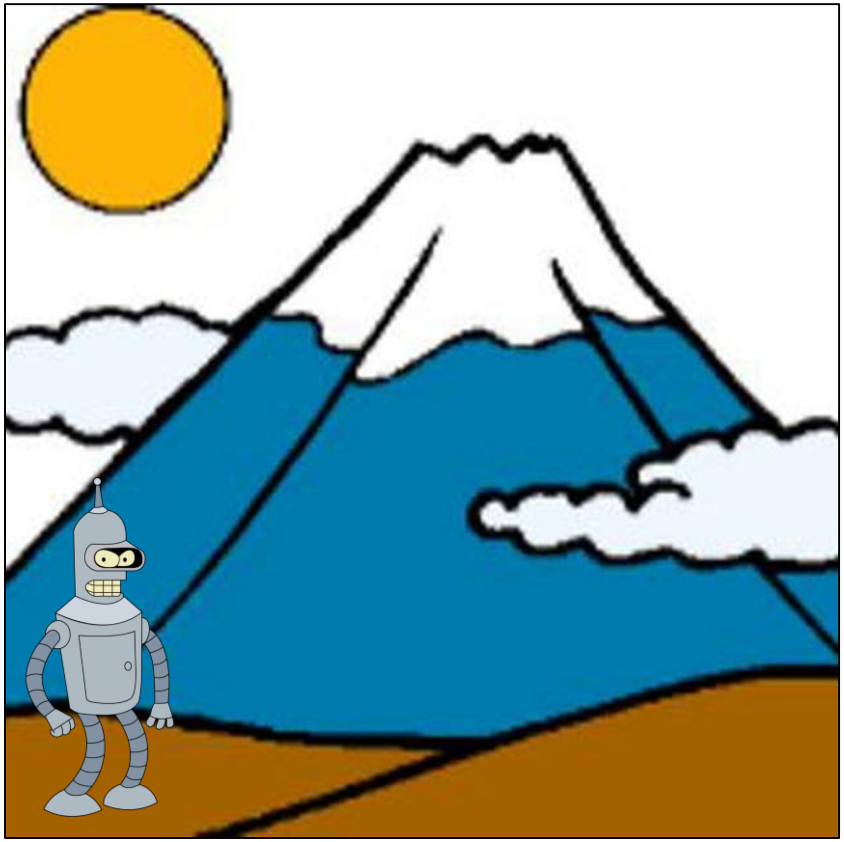
What is Optical Flow?

What is Optical Flow?

Movement

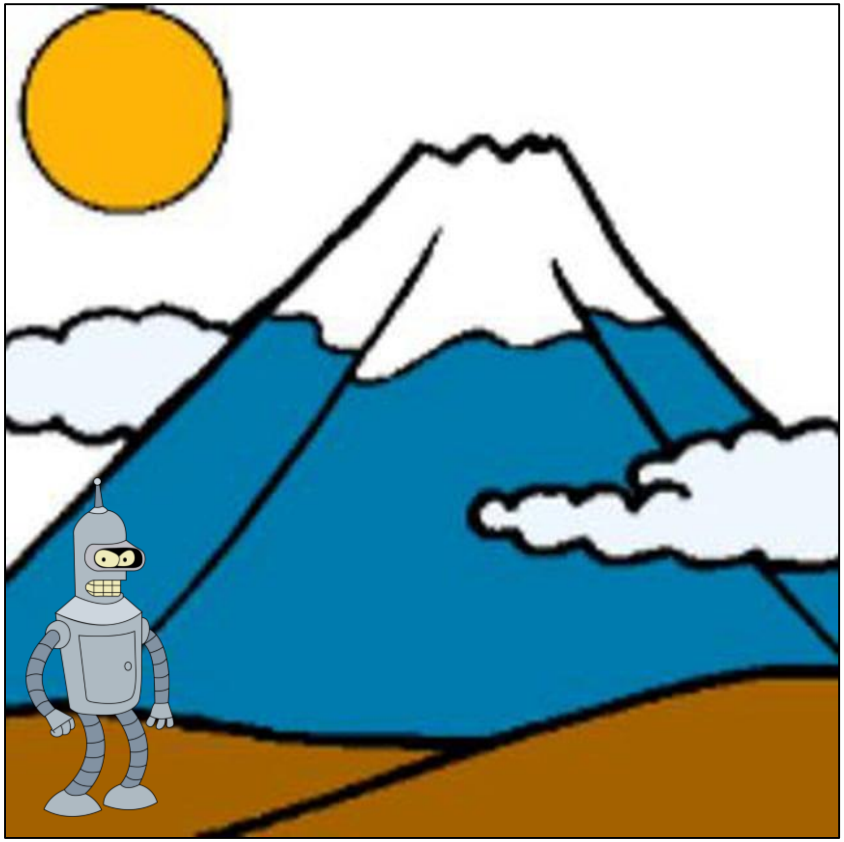
What is Optical Flow?

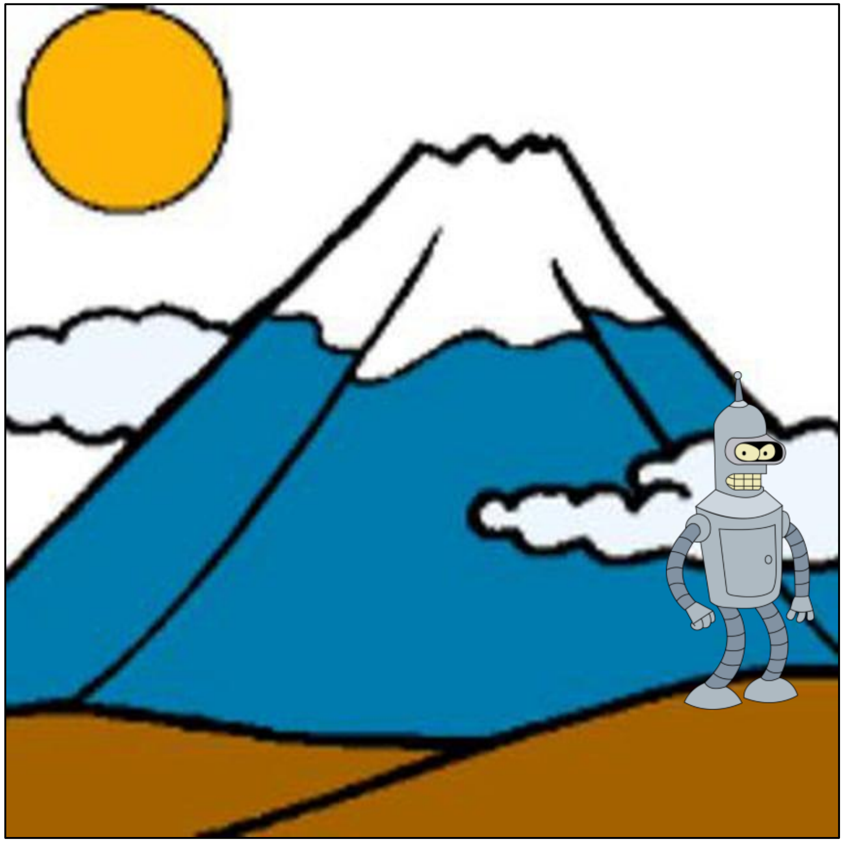
Movement



What is Optical Flow?

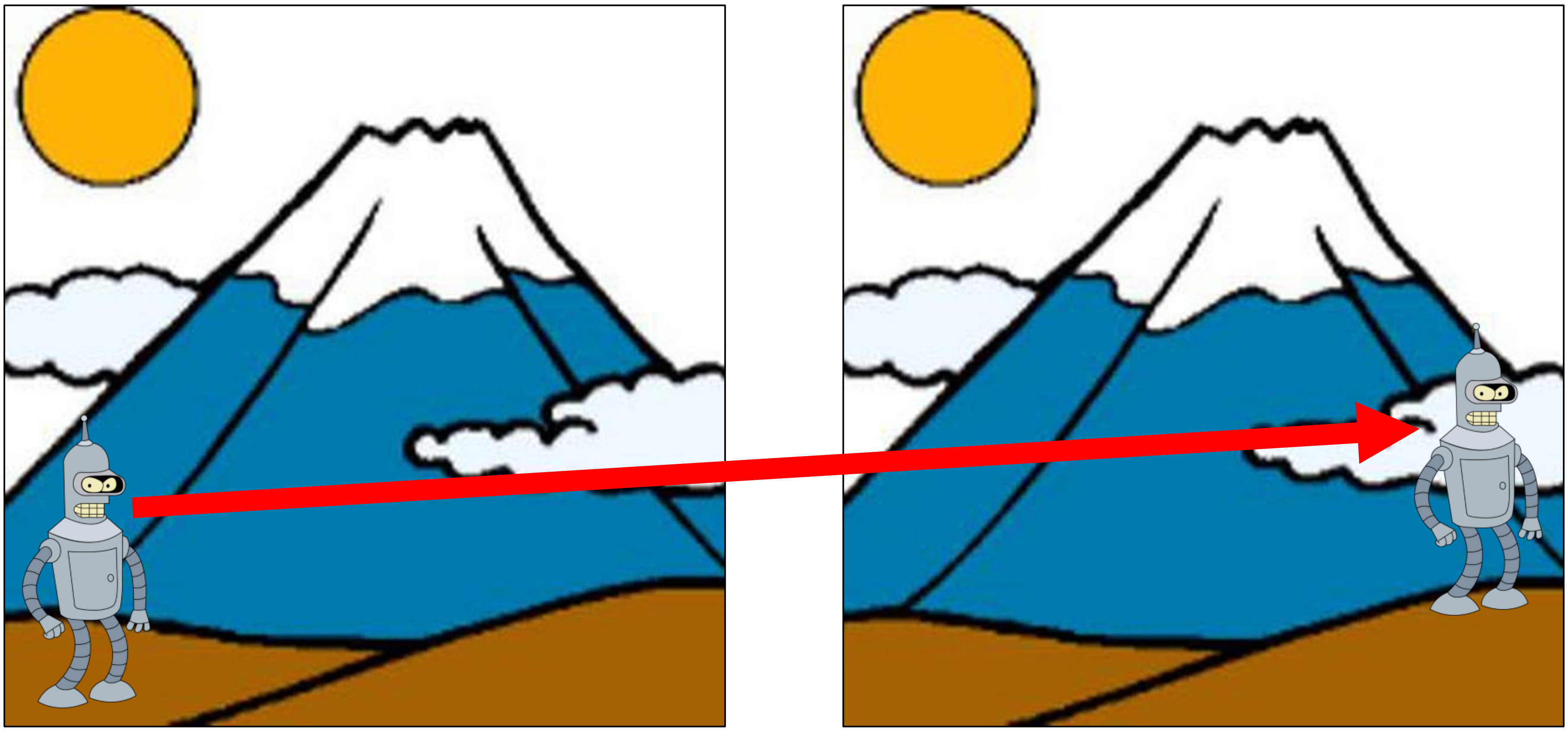
Movement



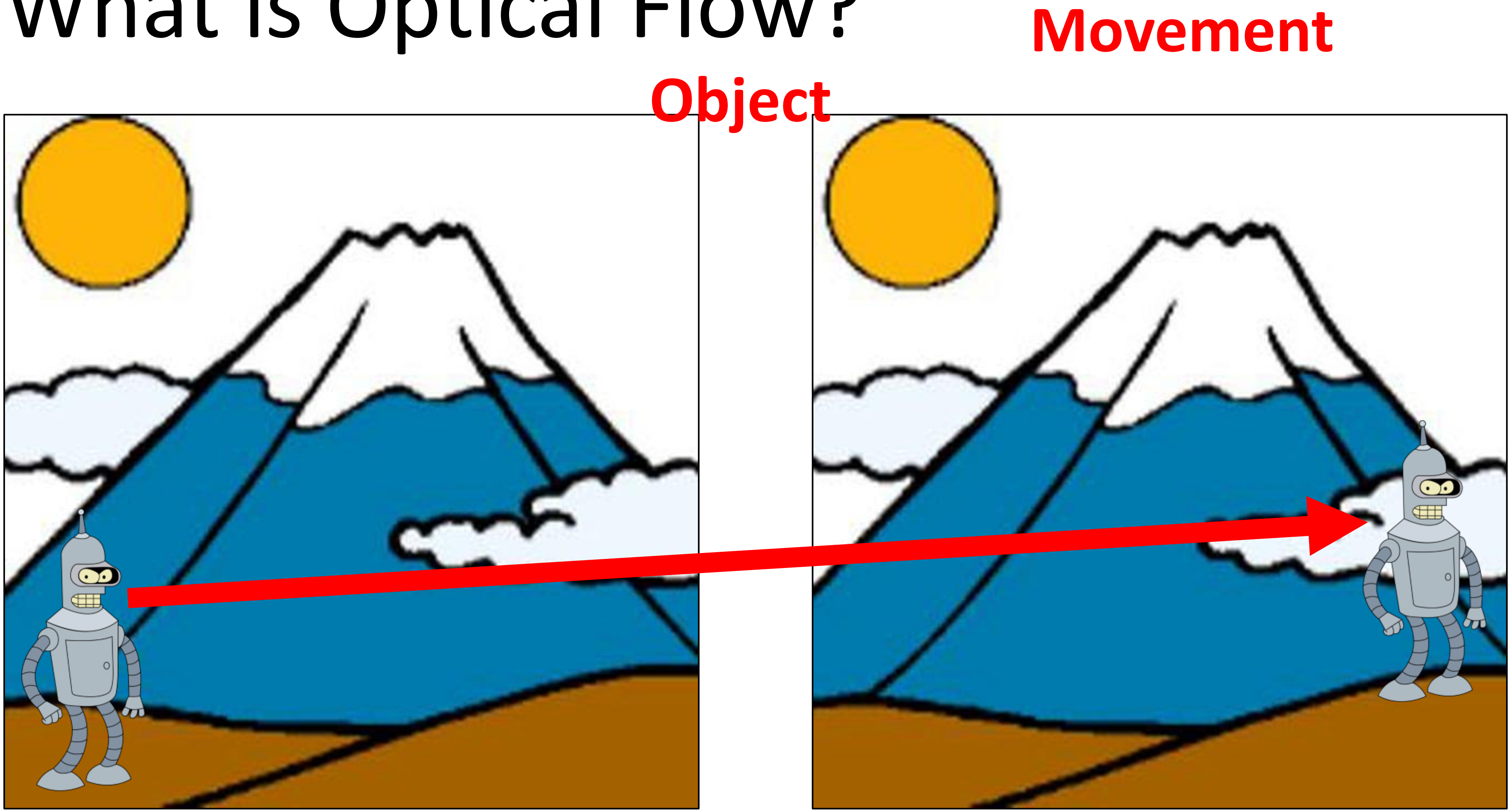


What is Optical Flow?

Movement



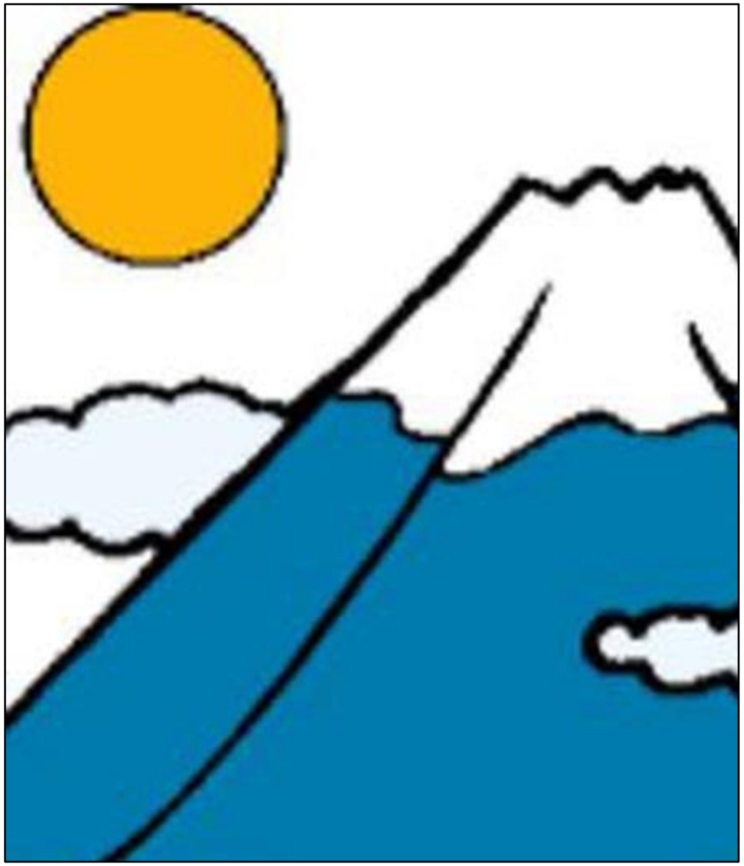
What is Optical Flow?

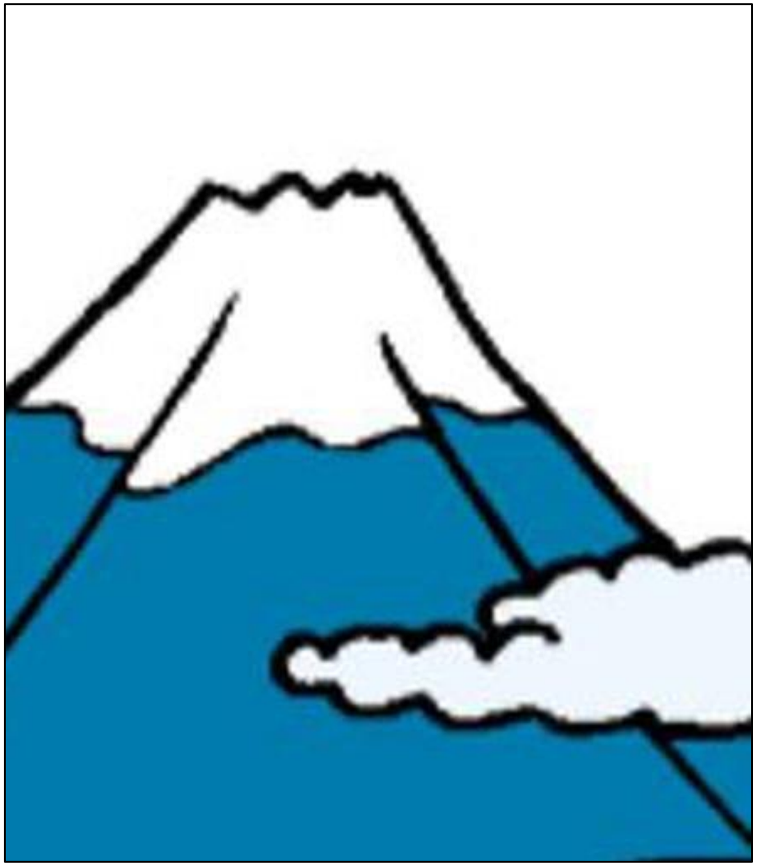


What is Optical Flow?

Pan

Movement

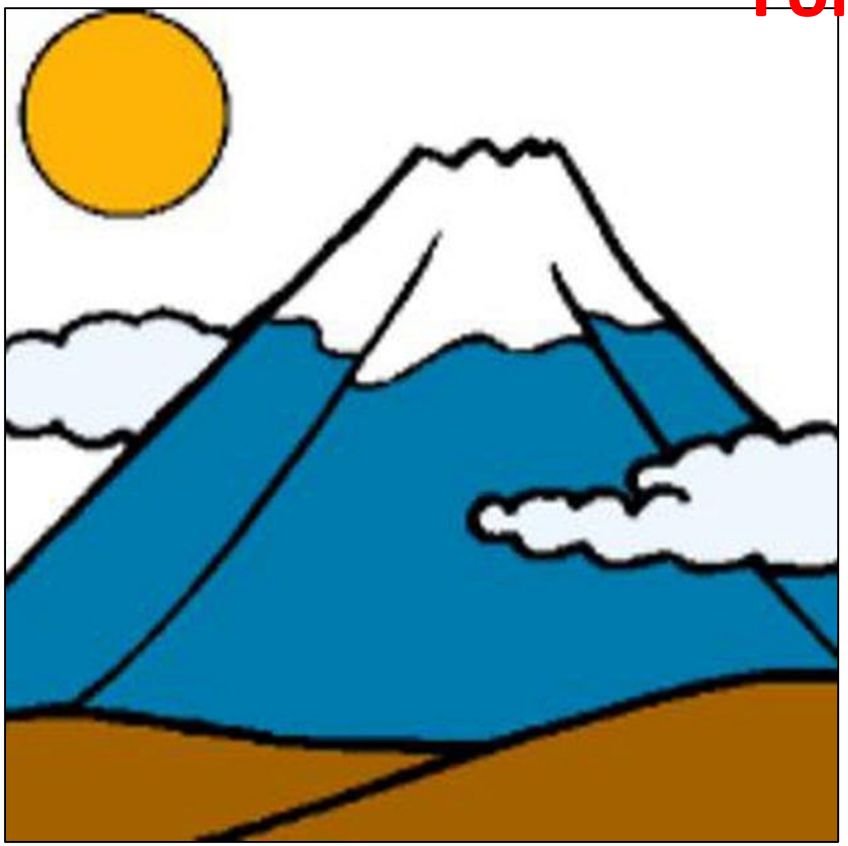


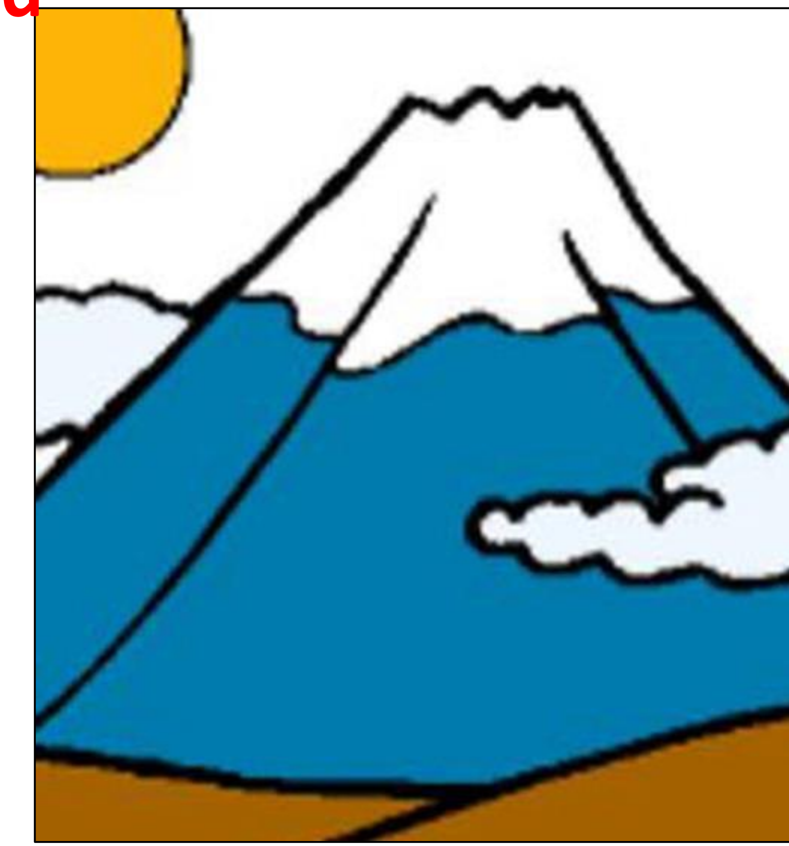


What is Optical Flow?

Movement

Forward





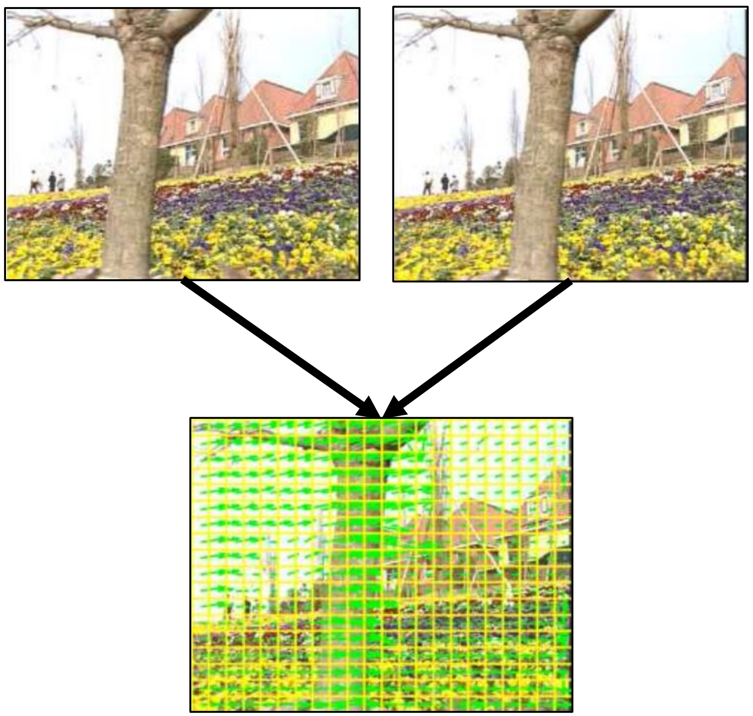
What is Optical Flow?

Movement

Why do we want Optical Flow?

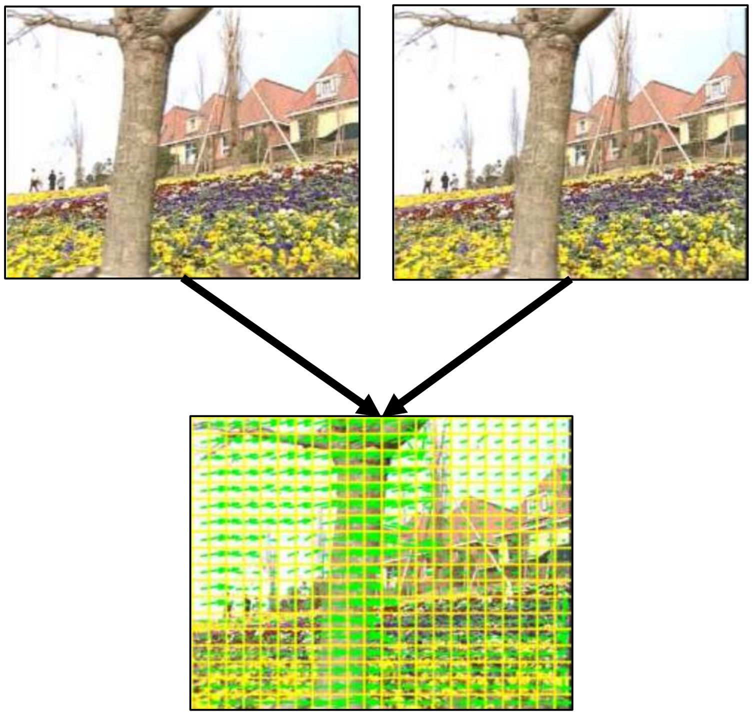
Why do we want Optical Flow?

Motion Estimation



Why do we want Optical Flow?

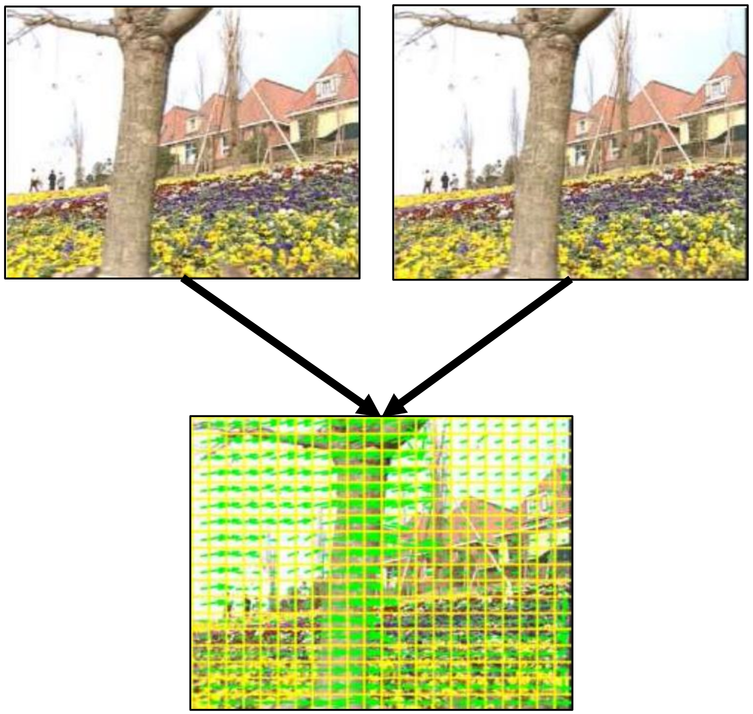
Motion Estimation



Object Tracking

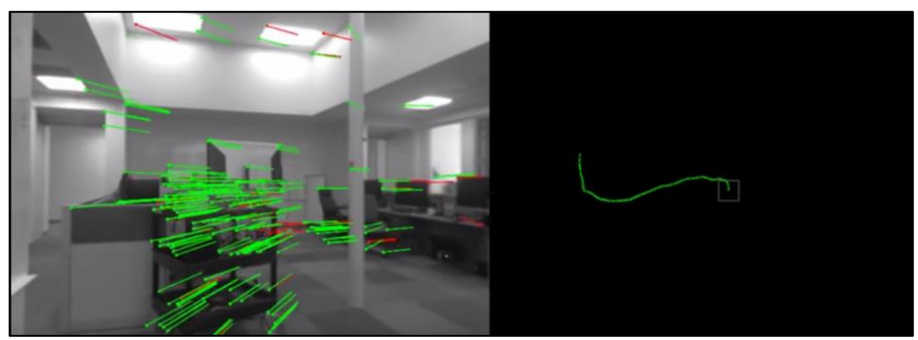
Why do we want Optical Flow?

Motion Estimation



Object Tracking

Visual Odometry

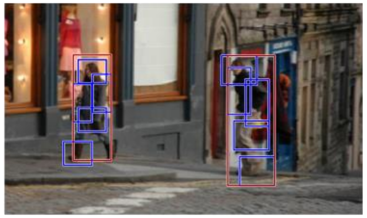


How do we find the flow in an image?

Feature Matching

Previously: Features!

- Highly descriptive local regions
- Ways to describe those regions
- Useful for:
    - Matching
    - Recognition
    - Detection



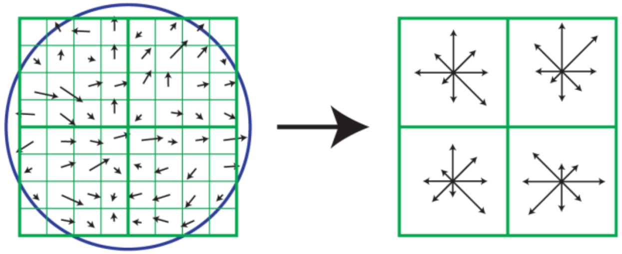
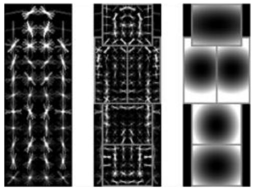
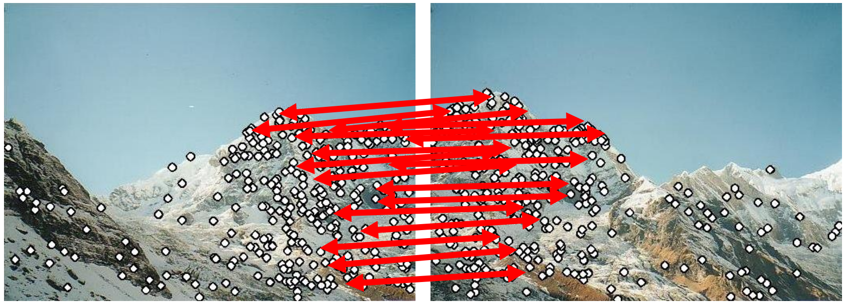


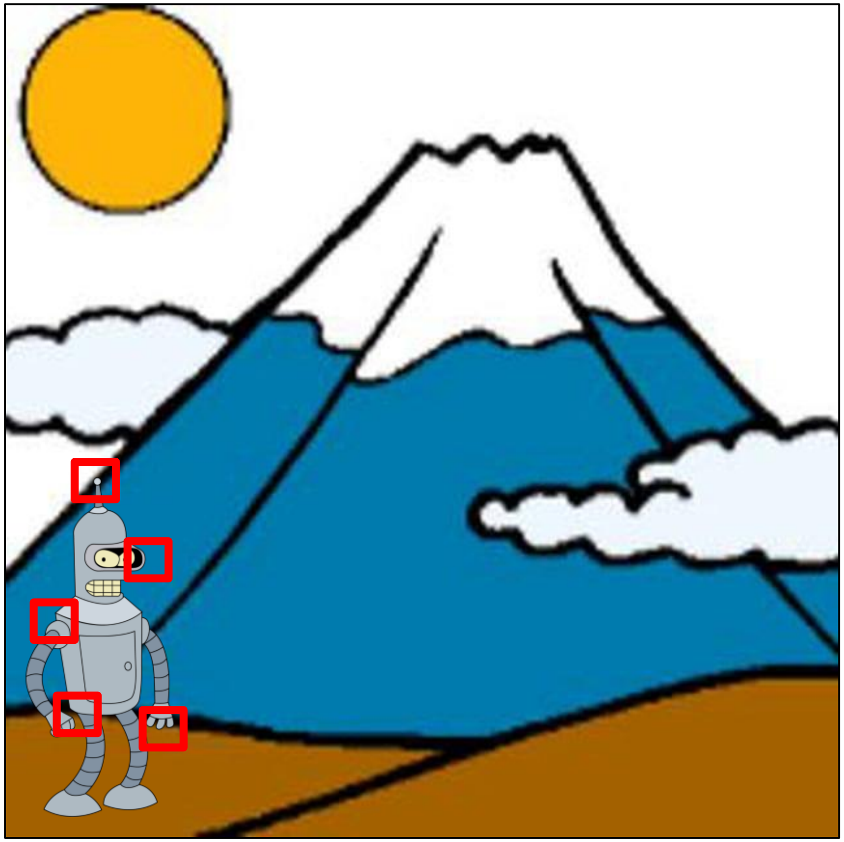
Image gradients

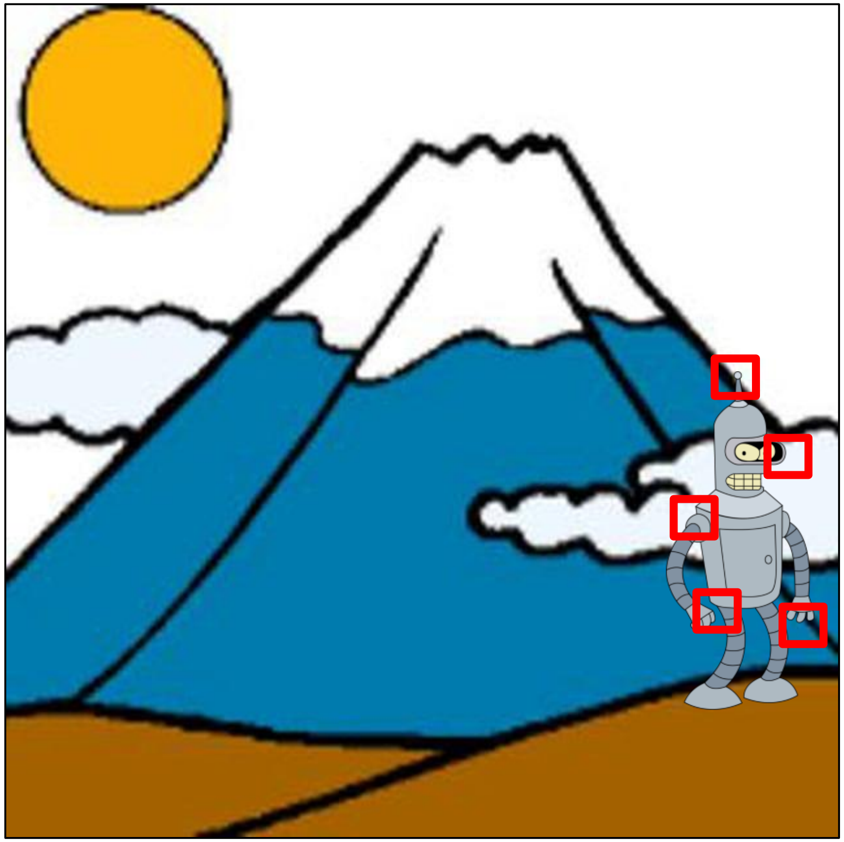
Keypoint descriptor



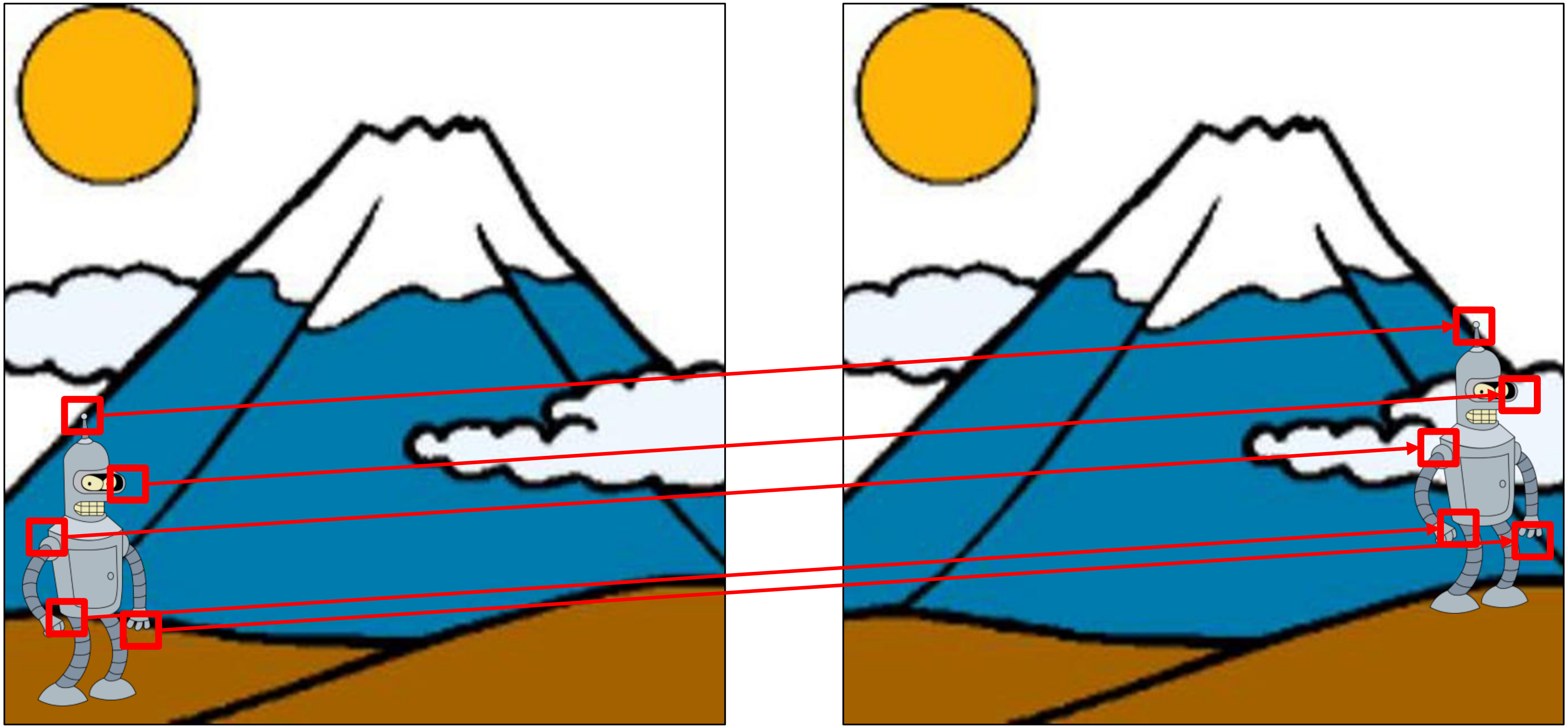


Feature Matching

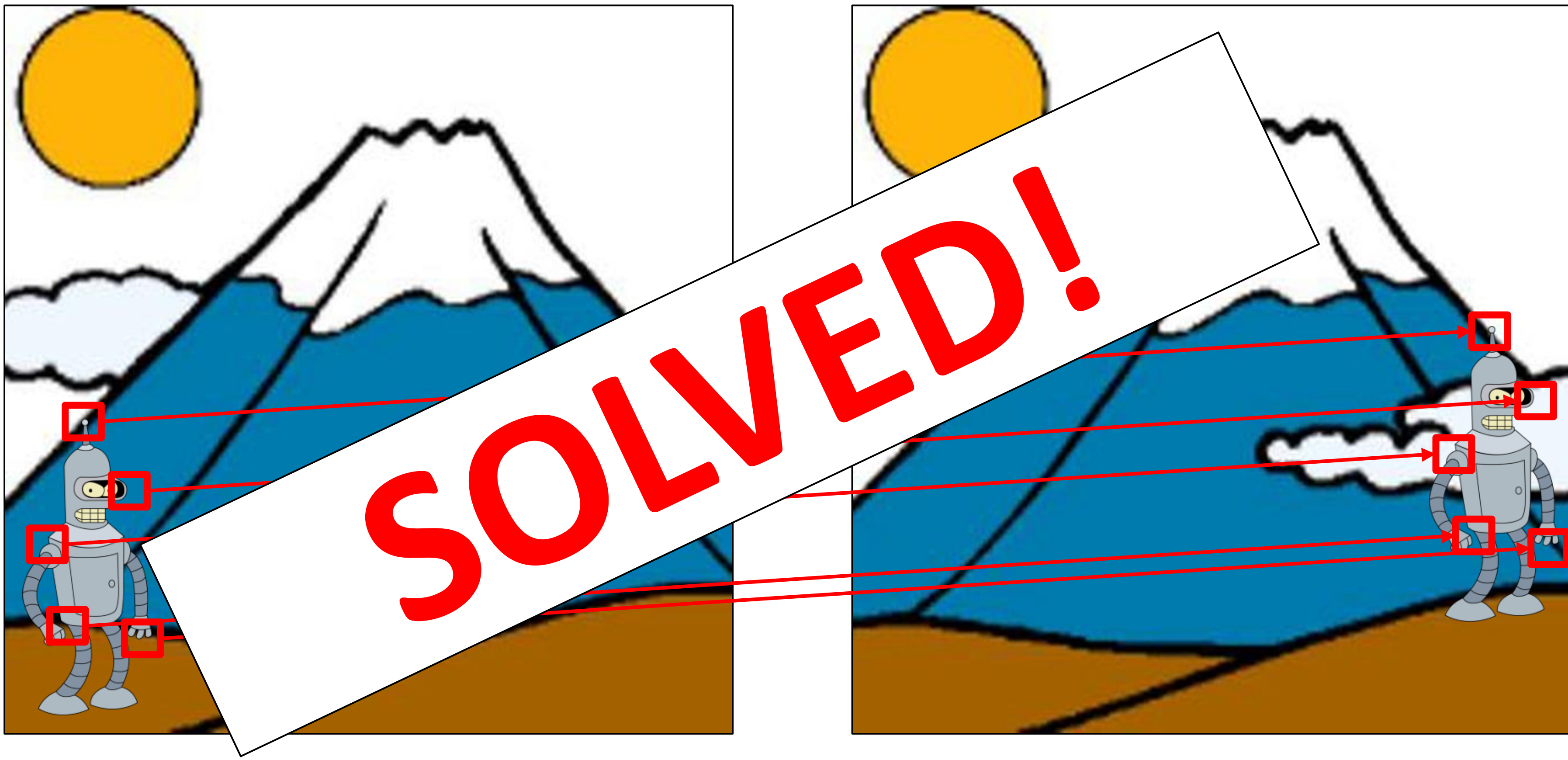




Feature Matching



Feature Matching



Feature Matching

Disadvantages:

Feature Matching

Disadvantages:

-Sparse!

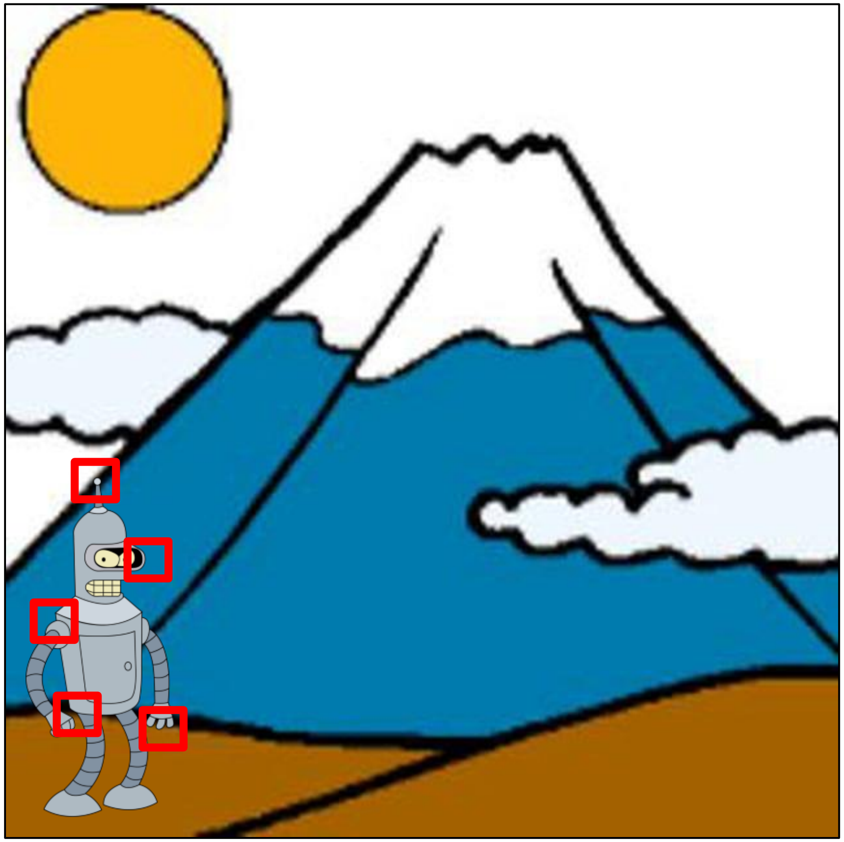
Feature Matching

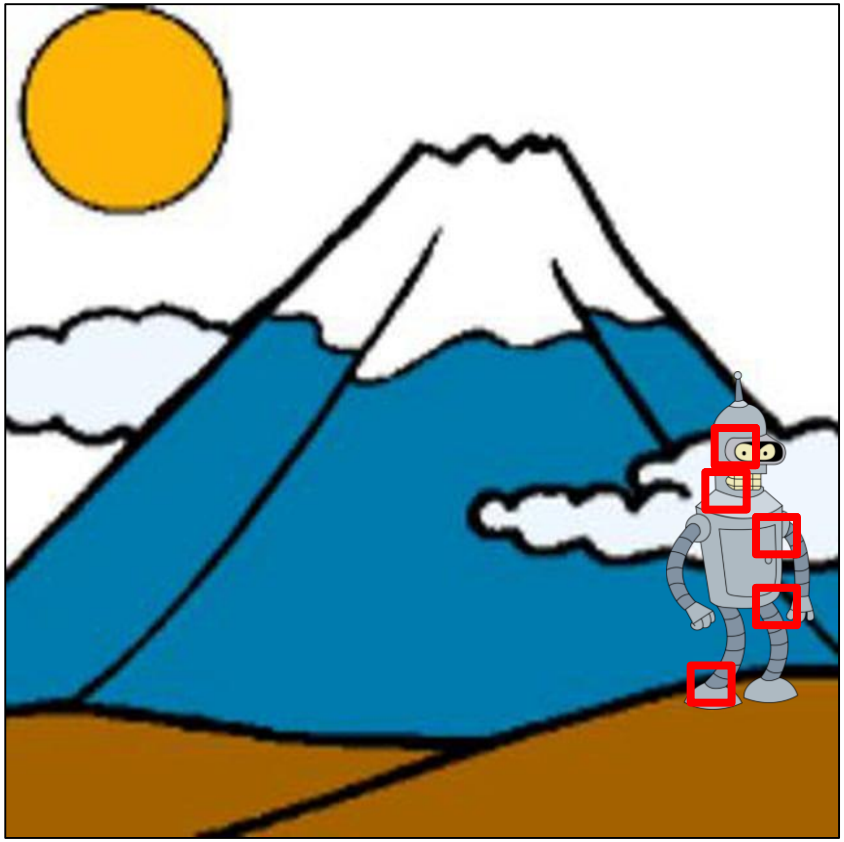
Disadvantages:

-Sparse!

-Feature alignment not exact

Feature Matching





Feature Matching

Disadvantages:

-Sparse!

-Feature alignment not exact

-Low accuracy

Feature Matching

Disadvantages:

-Sparse!

-Feature alignment not exact

-Low accuracy

Advantages:

Feature Matching

Disadvantages:

-Sparse!

-Feature alignment not exact

-Low accuracy

Advantages:

-Scale/rotation invariant

-*kinda* lighting invariant

-Can handle large movements

Feature Matching

Disadvantages:

-Sparse!

-Feature alignment not exact

-Low accuracy

Advantages:

-Scale/rotation invariant

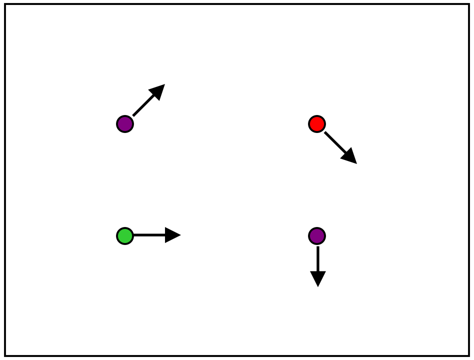
kinda lighting invariant

ge movements

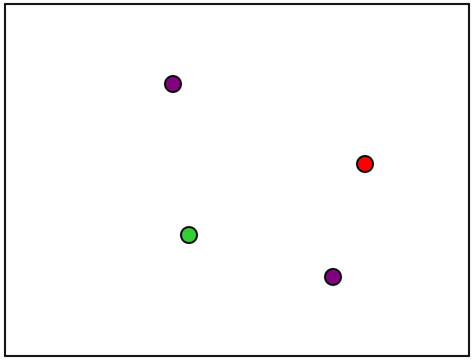
Overall: Doesn’t work very well for Optical Flow

What do we do instead?

Feature tracking



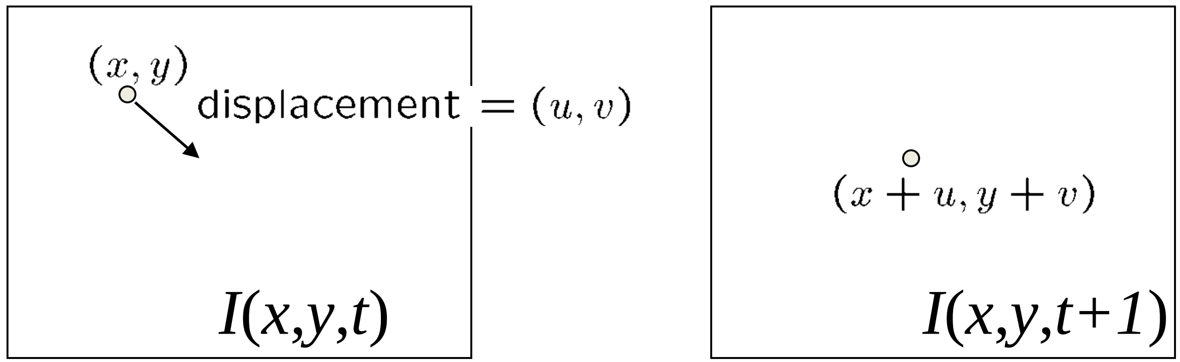
$I(x,y,t)$



$I(x,y,t+1)$

- Given two subsequent frames, estimate the point translation
- Key assumptions of Lucas-Kanade Tracker
  - **Brightness constancy:** projection of the same point looks the same in every frame
  - **Small motion:** points do not move very far
  - **Spatial coherence:** points move like their neighbors

The brightness constancy constraint



- Brightness Constancy Equation:

$$\boxed{I(x,y,t) = I(x+u, y+v, t+1)}$$

Take Taylor expansion of $I(x+u, y+v, t+1)$ at (x,y,t) to linearize the right side:

Image derivative along x   Difference over frames

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

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

- Difference in intensity at the same pixel between one image and the previous one.

The brightness constancy constraint

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

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

So: $$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

$$\rightarrow \nabla \mathrm{I} \cdot [\mathrm{u} \;\; \mathrm{v}]^{\mathrm{T}} + \mathrm{I_t} = 0$$

The brightness constancy constraint

Can we use this equation to recover image motion (u,v) at each pixel?

$$\nabla I \cdot [u \;\; v]^T + I_t = 0$$

- How many equations and unknowns per pixel?
    - One equation (this is a scalar equation!), two unknowns (u,v)

The component of the motion perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

Solving the ambiguity…

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of th* *International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- How to get more equations for a pixel?
- **Spatial coherence constraint**
- Assume the pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

Solving the ambiguity…

- Least squares problem:

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\times 2}{A} \; \underset{2\times 1}{d} = \underset{25\times 1}{b}$$

Matching patches across images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\times2}{A} \; \underset{2\times1}{d} = \underset{25\times1}{b}$$

Least squares solution for *d* given by $\boxed{(A^T A)\ d = A^T b}$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

The summations are over all pixels in the K x K window

$\boxed{d = (A^T A)^{-1} A^T b}$

Conditions for solvability

Optimal (u, v) satisfies Lucas-Kanade equation

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

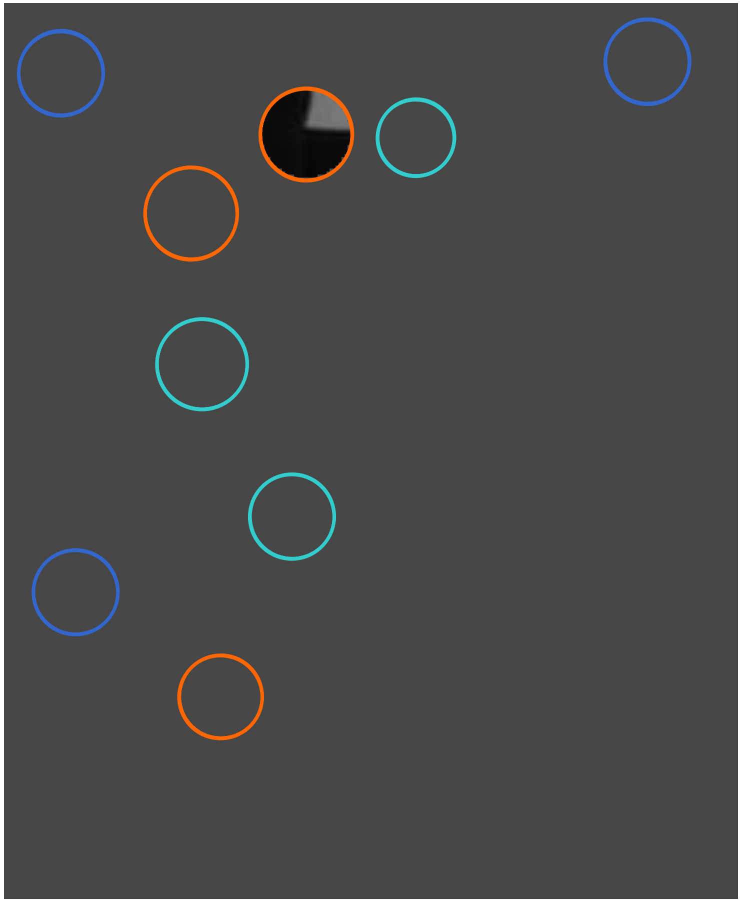
When is this solvable? I.e., what are good points to track?

- **A^TA** should be invertible
- **A^TA** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **A^TA** should not be too small
- **A^TA** should be well-conditioned
  - λ_1/λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind you of anything?

Criteria for Harris corner detector

Aperture problem

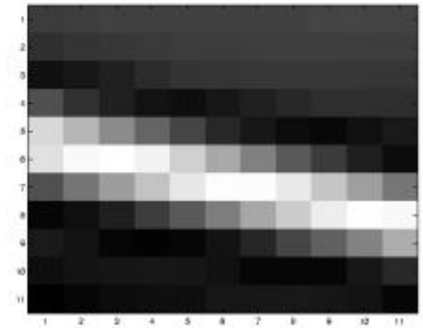


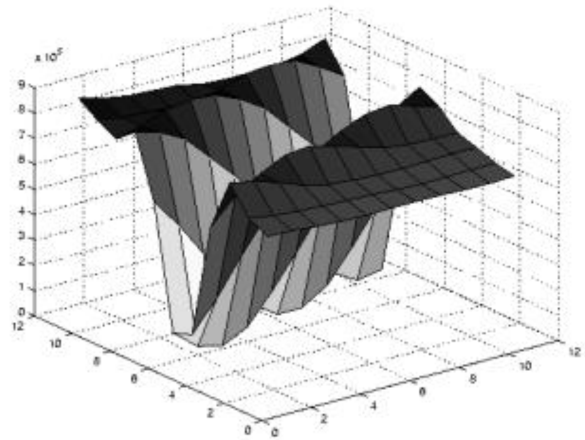
Corners

Lines

Flat regions

Edge



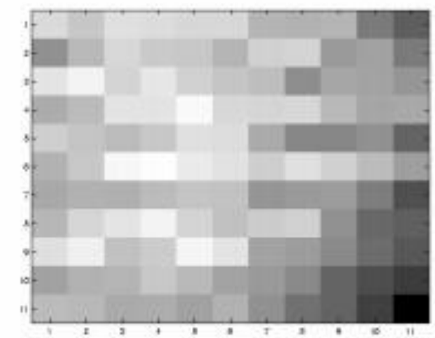


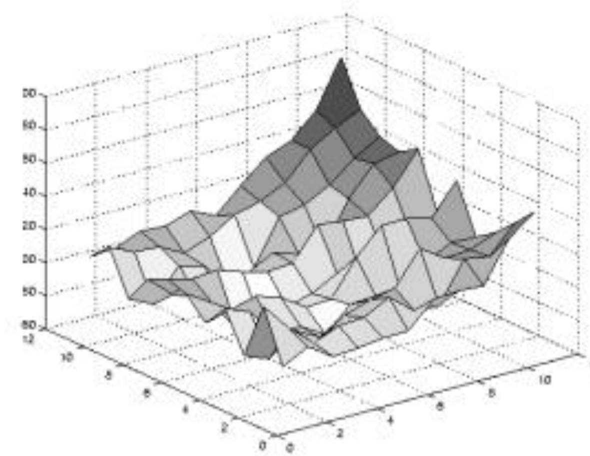
$$\sum \nabla I (\nabla I)^T$$

– large gradients, all the same

– large λ_1, small λ_2

Low Texture Region

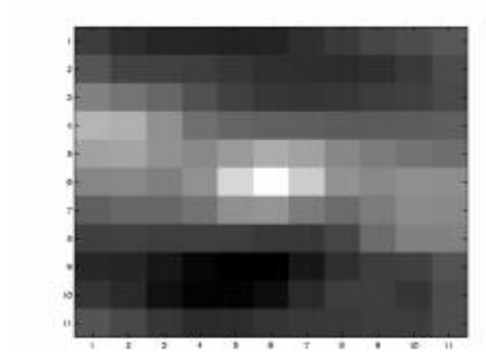


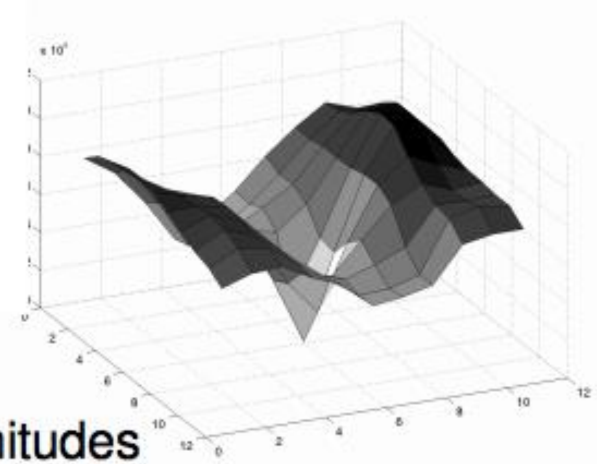


$$\sum \nabla I (\nabla I)^T$$

- gradients have small magnitude
- small λ_1, small λ_2

High Texture Region





$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes

– large λ_1, large λ_2

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose A^TA is easily invertible
  - Suppose there is not much noise in the image

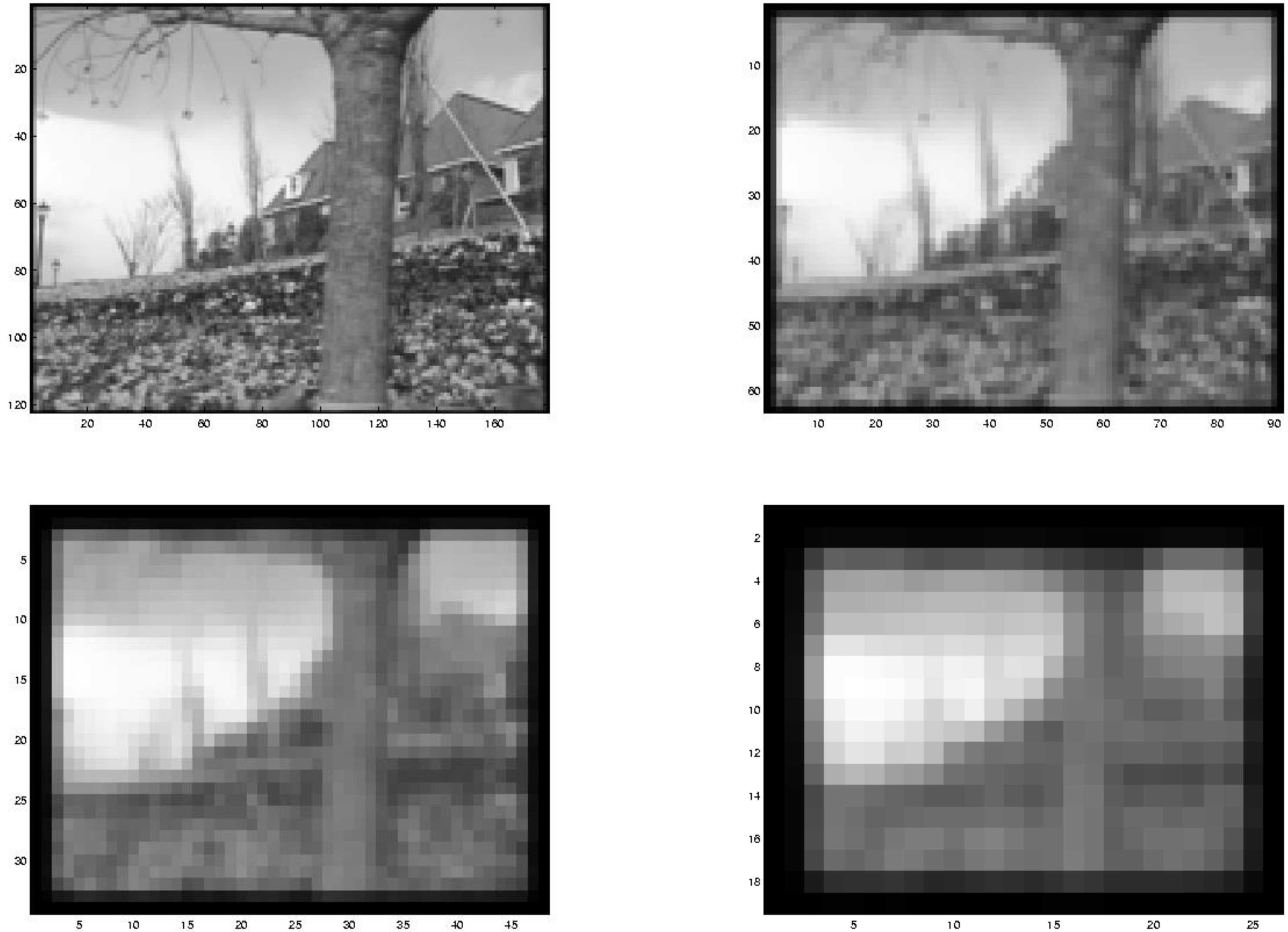
When our assumptions are violated

- Brightness constancy is **not** satisfied
- The motion is **not** small
- A point does **not** move like its neighbors
  - window size is too large
  - what is the ideal window size?

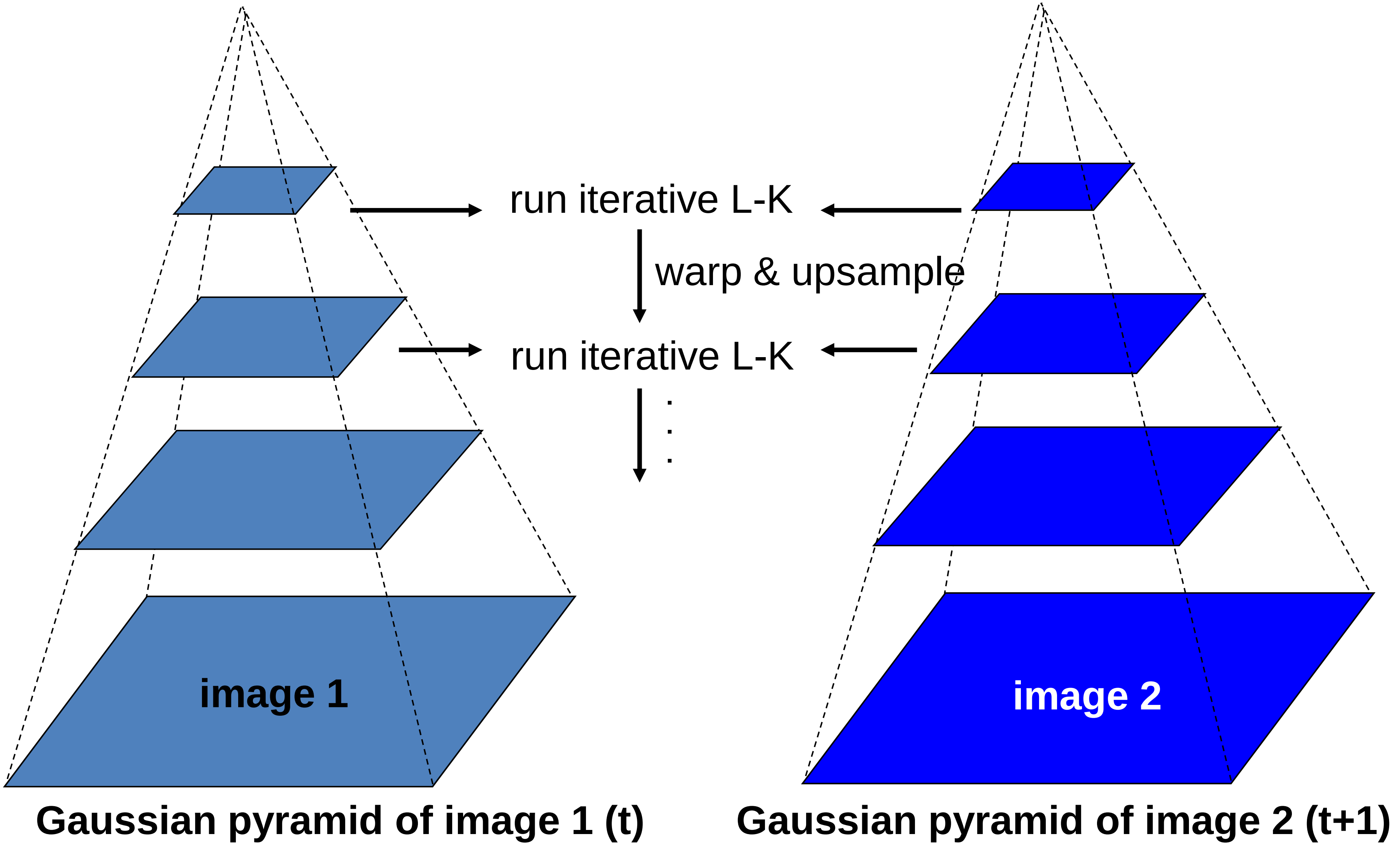
Revisiting the small motion assumption

- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
  - How might we solve this problem?

Reduce the resolution!



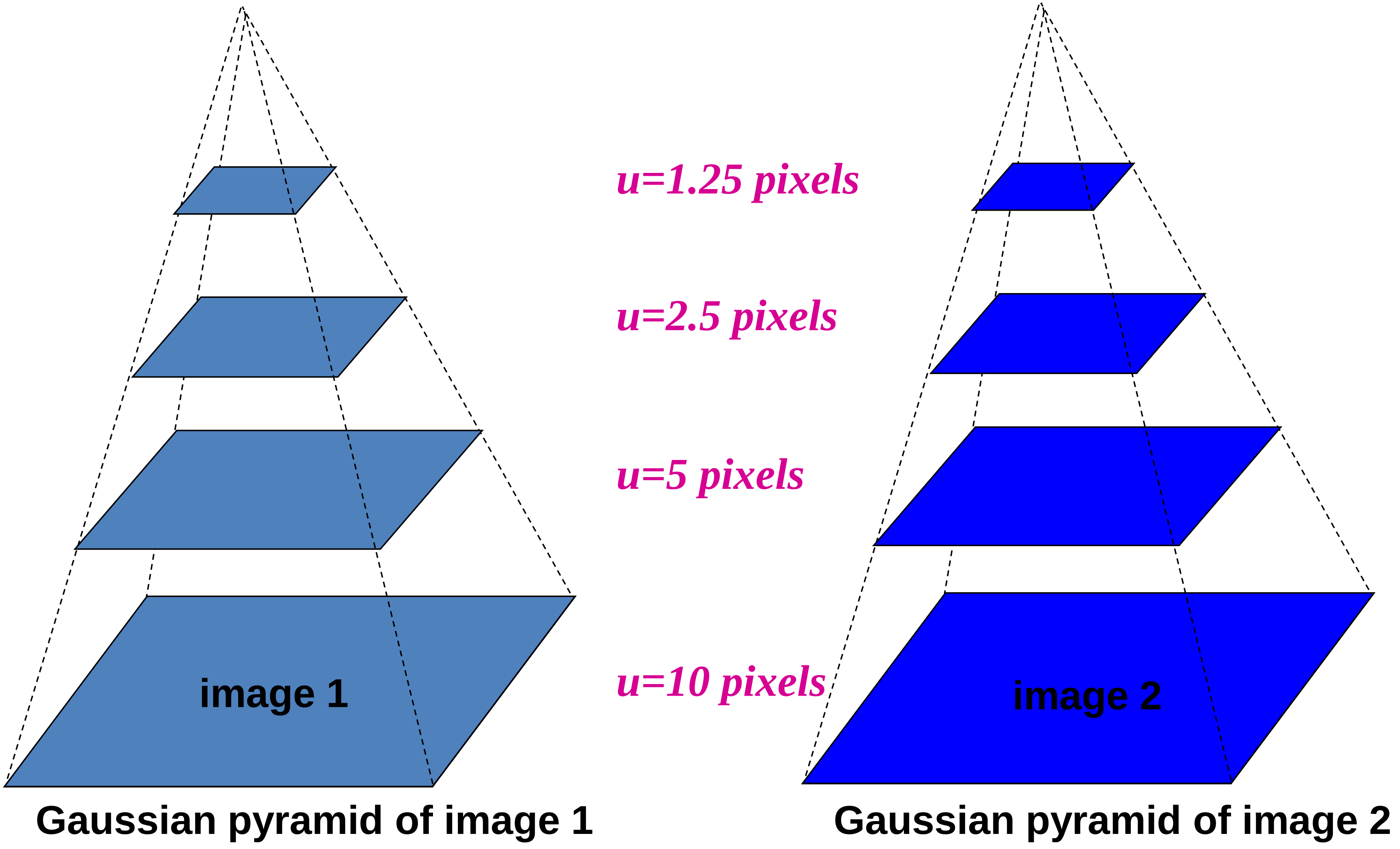
Coarse-to-fine optical flow estimation



A Few Details

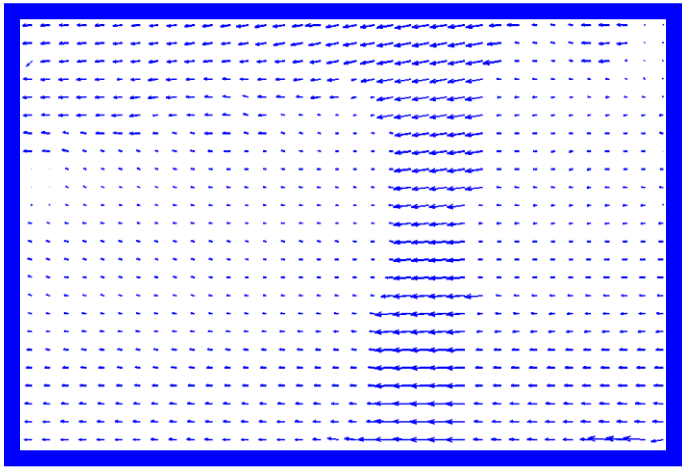
- Top Level
  - Apply L-K to get a flow field representing the flow from the first frame to the second frame.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K on the new warped image to get a flow field from it to the second frame.
  - Repeat till convergence.
- Next Level
  - Upsample the flow field to the next level as the first guess of the flow at that level.
  - Apply this flow field to warp the first frame toward the second frame.
  - Rerun L-K and warping till convergence as above.
- Etc.

Coarse-to-fine optical flow estimation

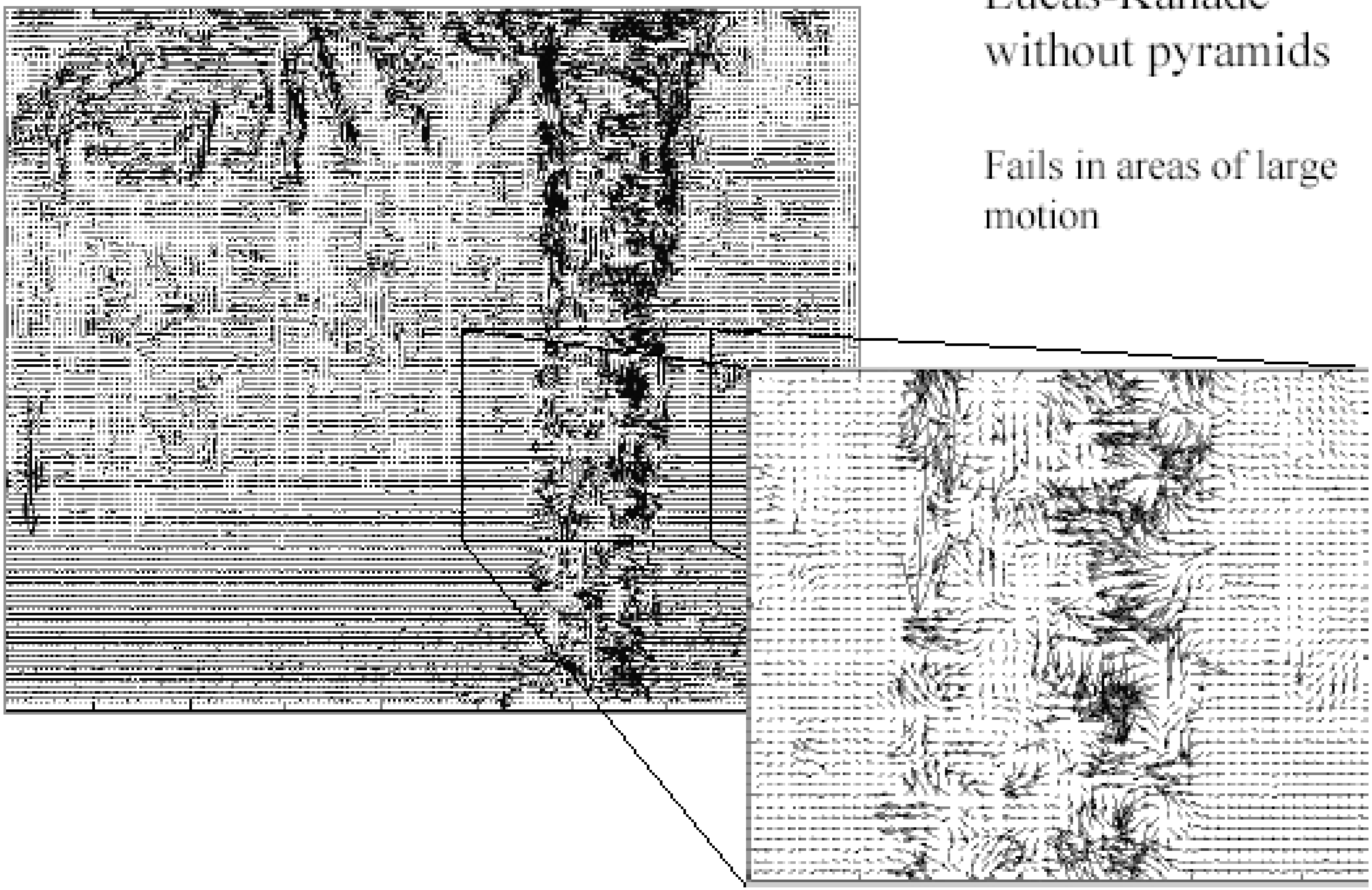


The Flower Garden Video

What should the
optical flow be?



Optical Flow Results

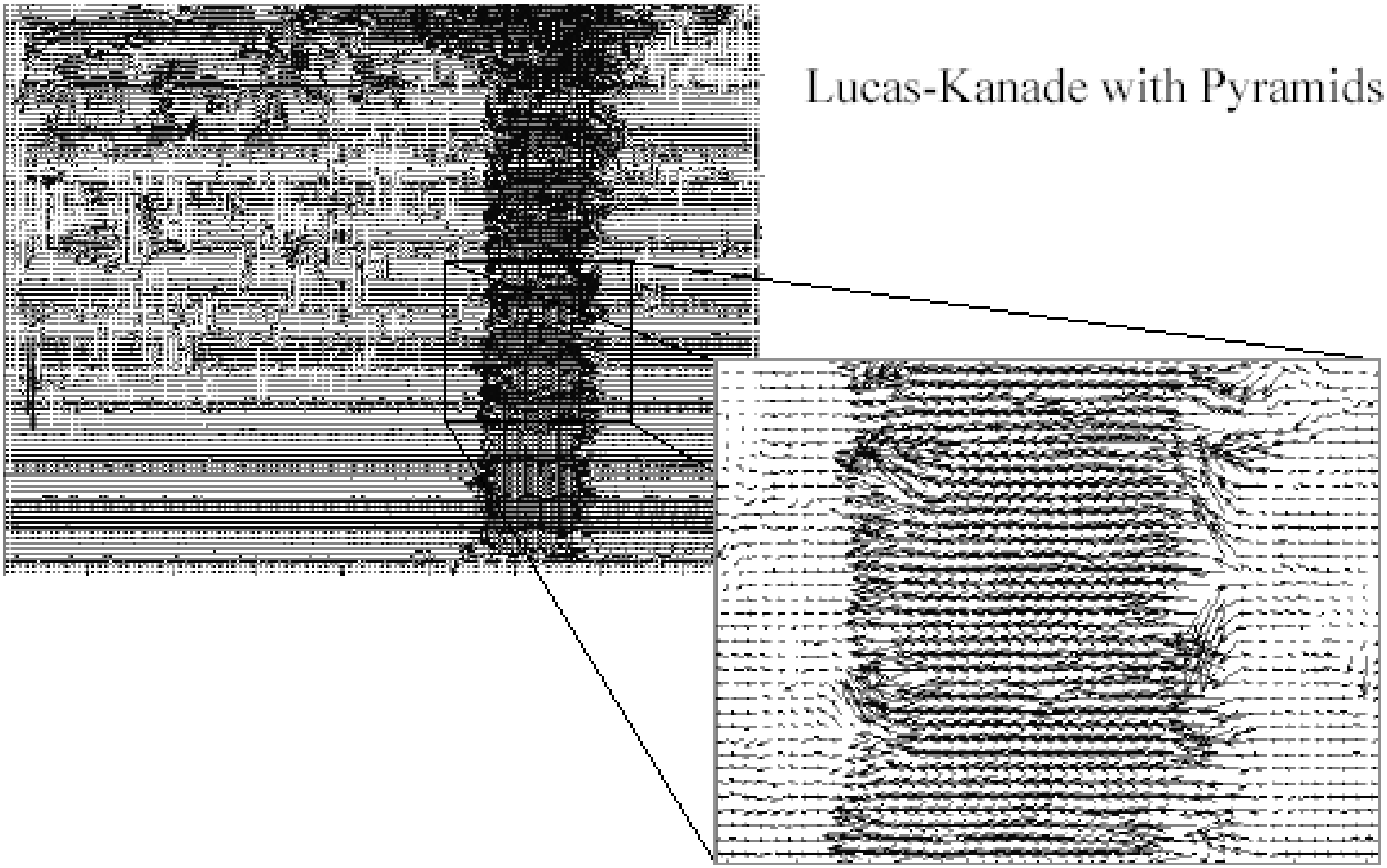


Lucas-Kanade without pyramids

Fails in areas of large motion

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Optical Flow Results



Lucas-Kanade with Pyramids

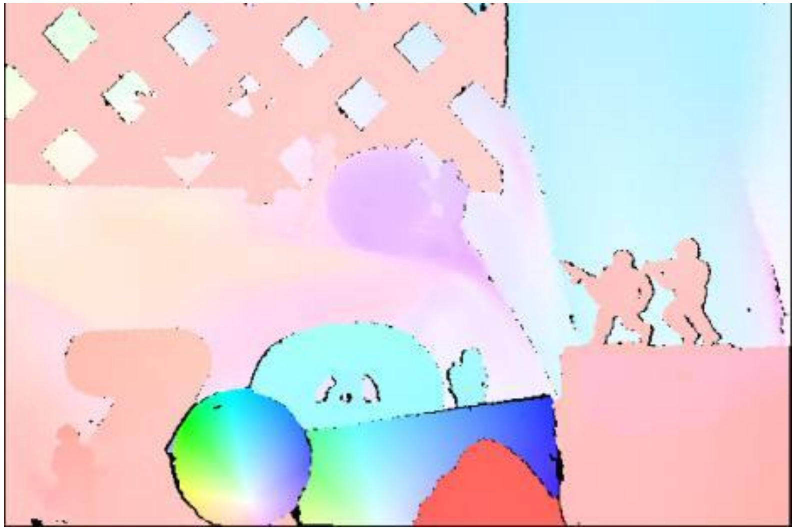
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Flow quality evaluation

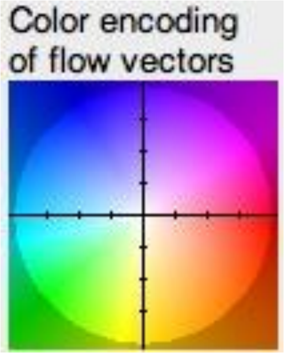
Flow quality evaluation

Flow quality evaluation

- Middlebury flow page
  - http://vision.middlebury.edu/flow/

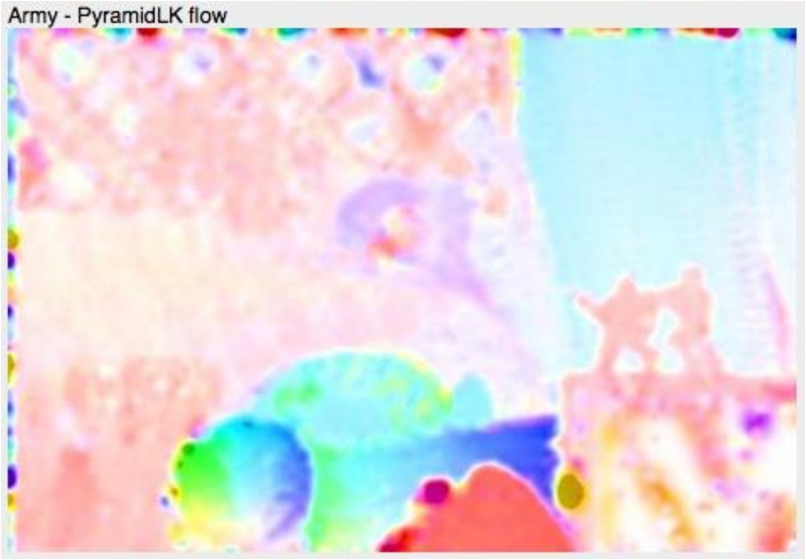


Ground Truth

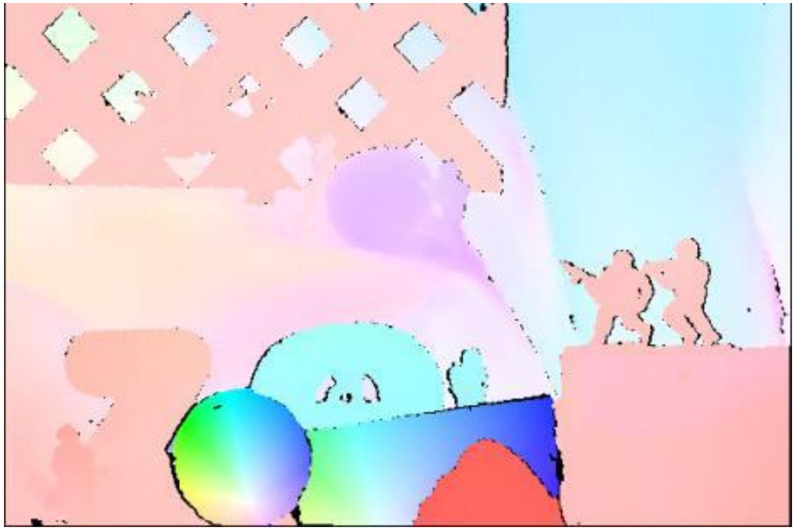


Flow quality evaluation

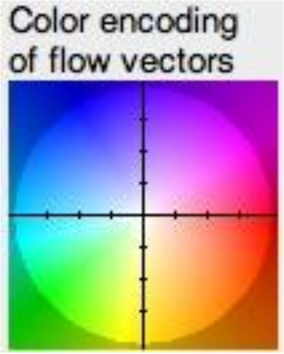
- Middlebury flow page
  - http://vision.middlebury.edu/flow/



Lucas-Kanade flow



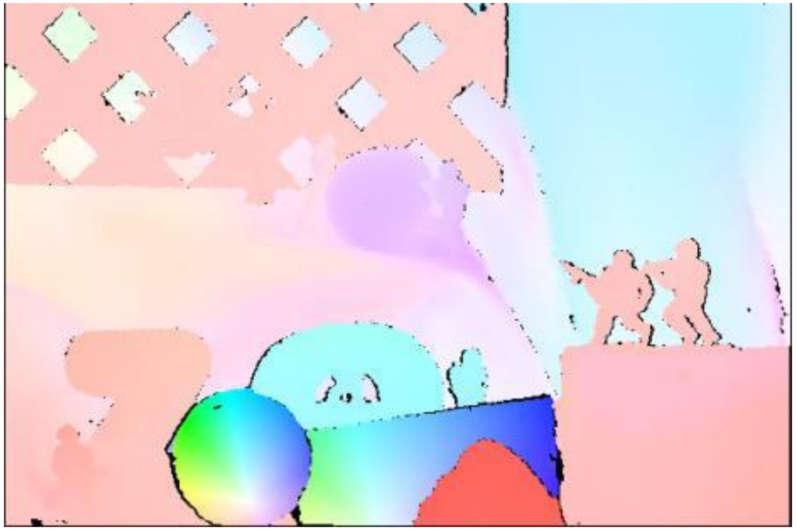
Ground Truth



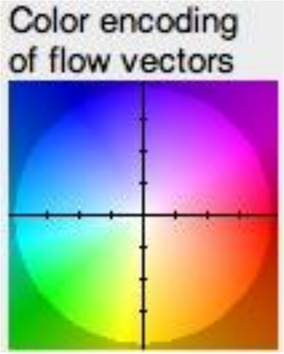
Flow quality evaluation

- Middlebury flow page
  - http://vision.middlebury.edu/flow/

Best-in-class alg



Ground Truth



Video stabilization

Video denoising

Video super resolution

Robust Visual Motion Analysis:
Piecewise-Smooth Optical Flow

Ming Ye
Electrical Engineering
University of Washington

Estimating Piecewise-Smooth Optical Flow with Global Matching and Graduated Optimization

Problem Statement:

Assuming only brightness conservation and piecewise-smooth motion, find the optical flow to best describe the intensity change in three frames.

Approach: Matching-Based Global Optimization

- **Step 1. Robust local gradient-based method for high-quality initial flow estimate.**

  Uses least median of squares instead of regular least squares.

- **Step 2. Global gradient-based method to improve the flow-field coherence.**

  Minimizes a global energy function $E = \Sigma\,(E_B(V_i) + E_S(V_i))$ where E_B is the brightness difference and E_S is the smoothness at flow vector V_i

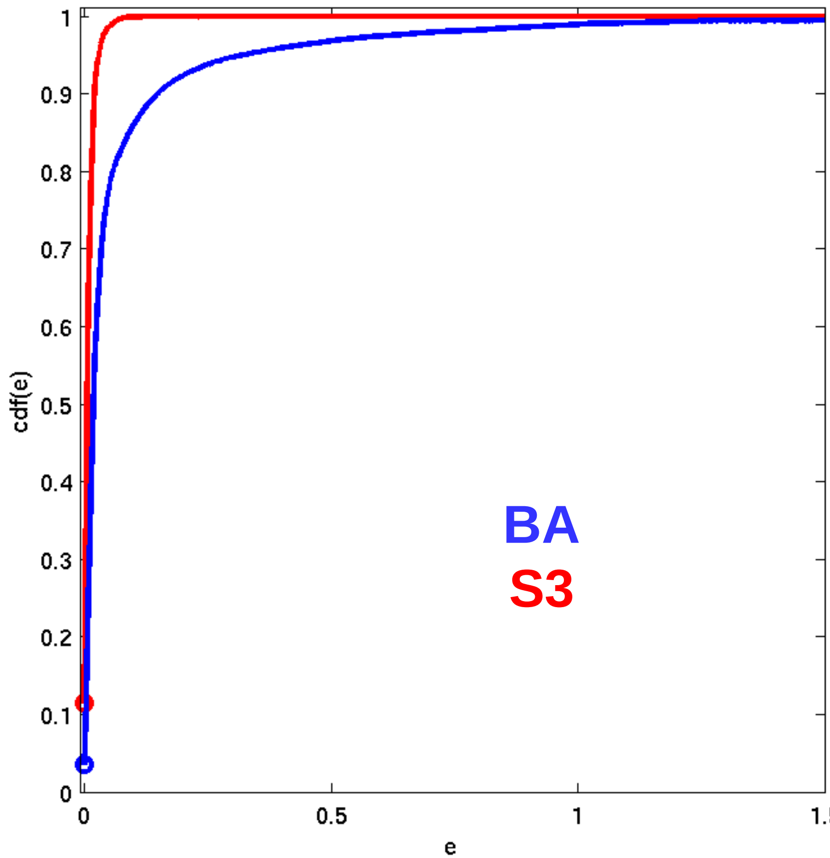
- **Step 3. Global matching that minimizes energy by a greedy approach.**

  Visits each pixel and updates it to be consistent with neighbors, iteratively.

TT: Translating Tree

150x150 (Barron 94)

	$e_{\angle}(^\circ)$	$e_{\lvert\bullet\rvert}(\text{pix})$	$\bar{e}(\text{pix})$
BA	2.60	0.128	0.0724
S3	0.248	0.0167	0.00984

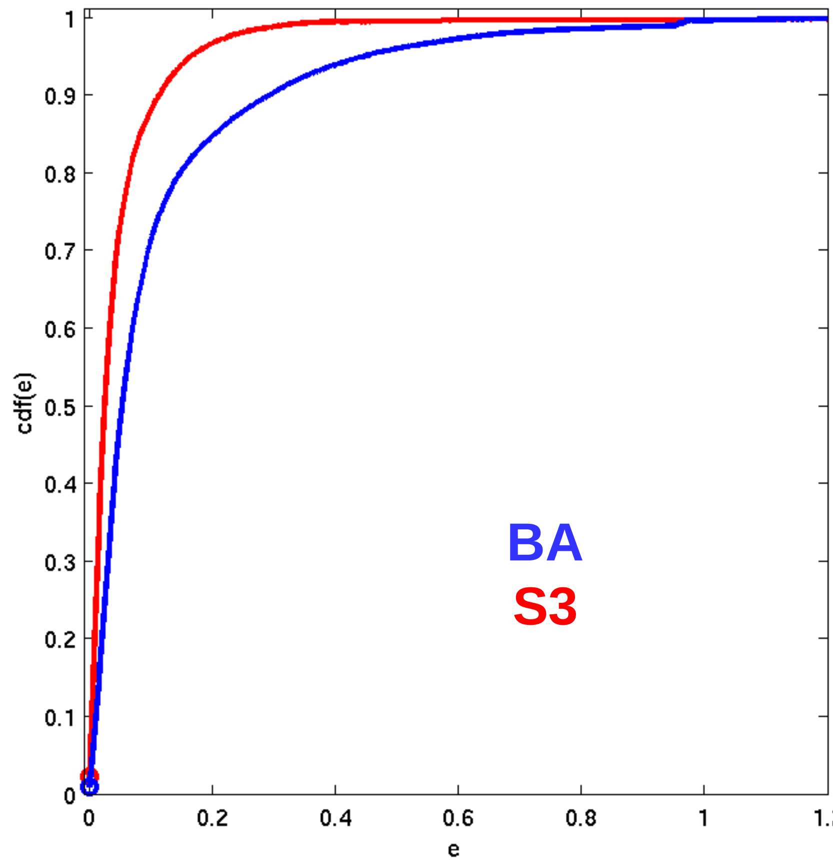


e: error in pixels, cdf: culmulative distribution function for all pixels

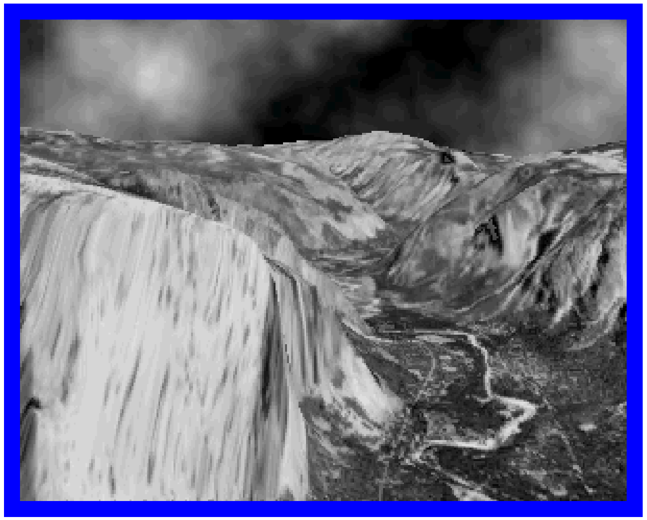
DT: Diverging Tree

150x150 (Barron 94)

	$e_{\angle}(^\circ)$	$e_{\|\bullet\|}(\mathrm{pix})$	$\bar{e}(\mathrm{pix})$
BA	6.36	0.182	0.114
S3	2.60	0.0813	0.0507

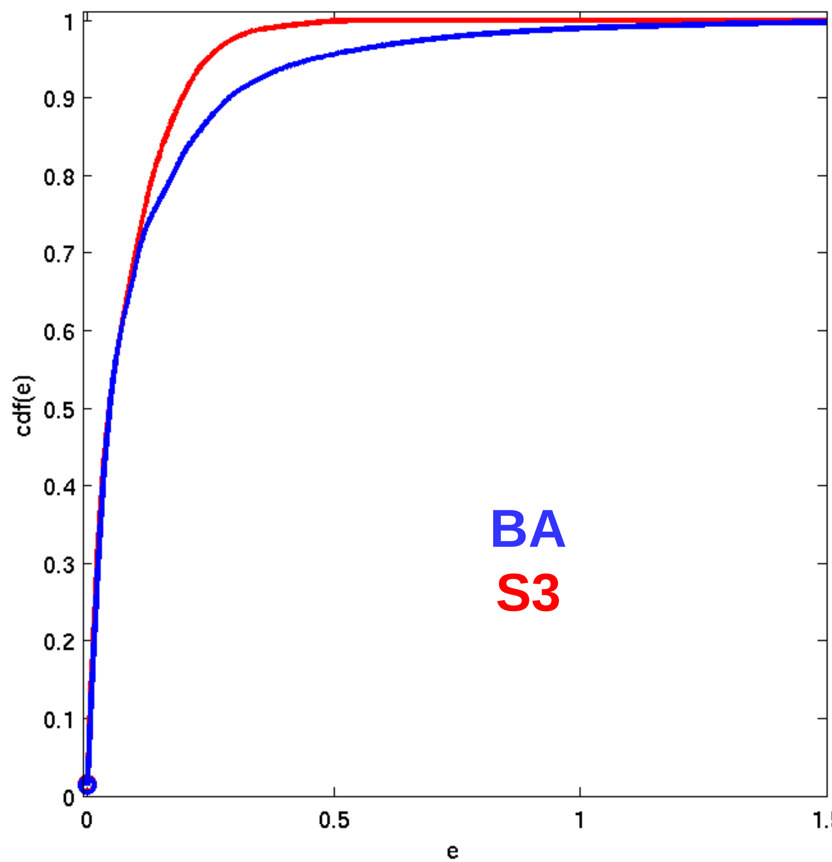


YOS: Yosemite Fly-Through



316x252 (Barron, cloud excluded)

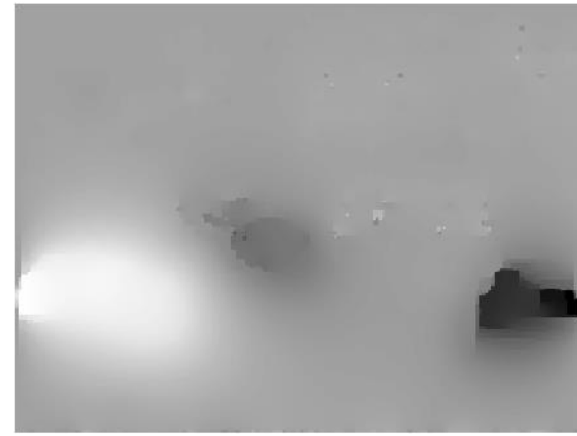
	$e_{\angle}(^\circ)$	$e_{\lvert\bullet\rvert}(\text{pix})$	$\bar{e}(\text{pix})$
BA	**2.71**	**0.185**	**0.118**
S3	**1.92**	**0.120**	**0.0776**



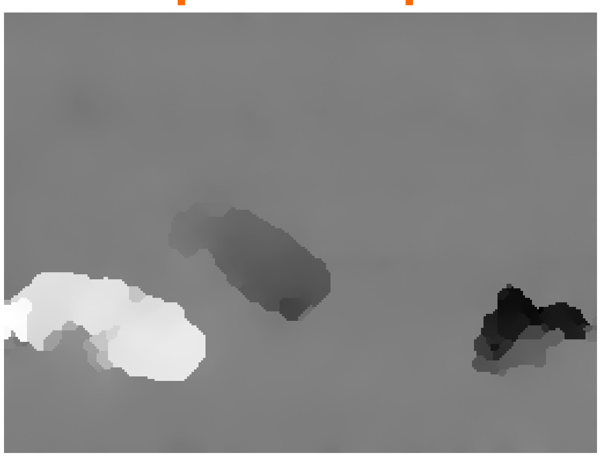
TAXI: Hamburg Taxi

256x190, (Barron 94)
max speed 3.0 pix/frame

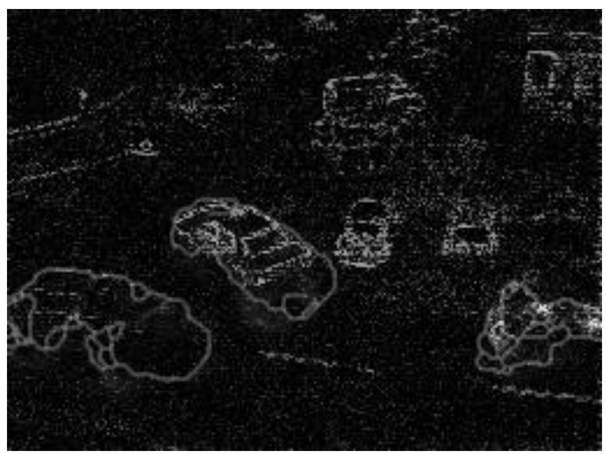
LMS



BA



Ours

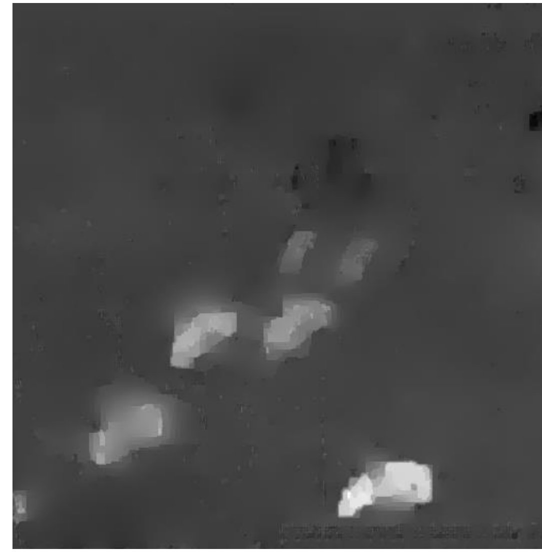


Error map

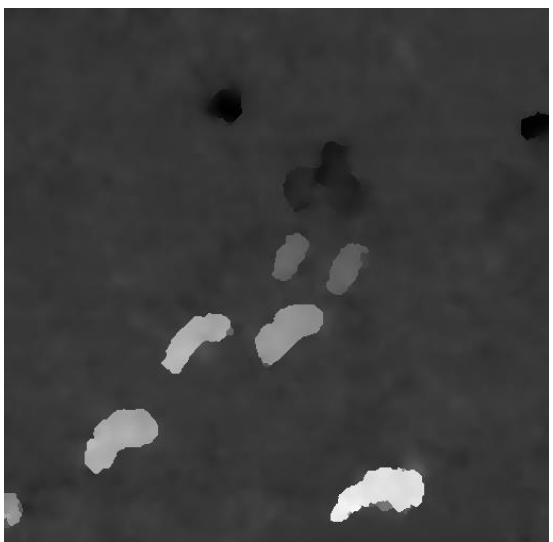
Smoothness error

Traffic

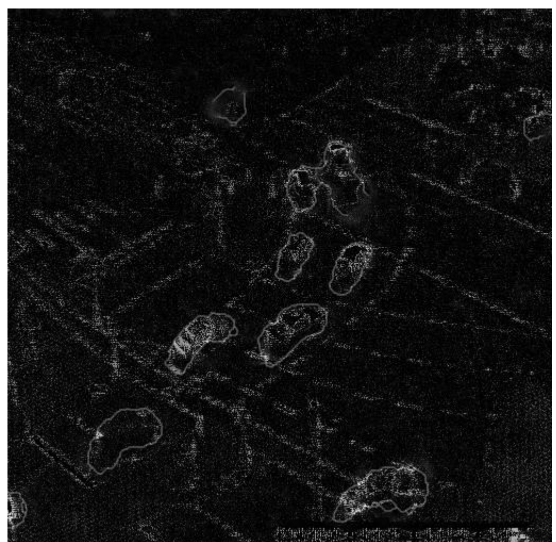
512x512
(Nagel)
max speed:
6.0 pix/frame



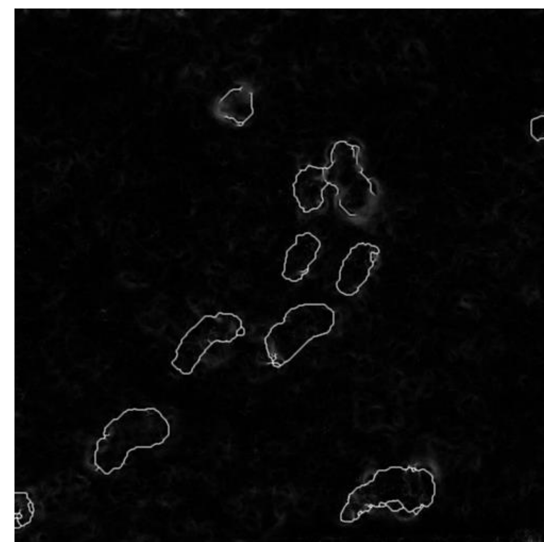
BA



Ours



Error map

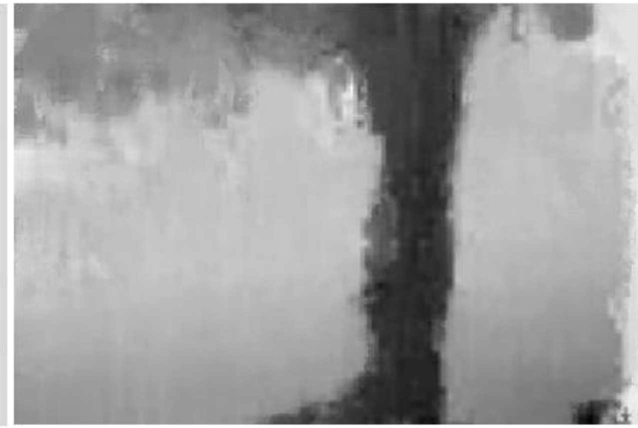


Smoothness error

FG: Flower Garden

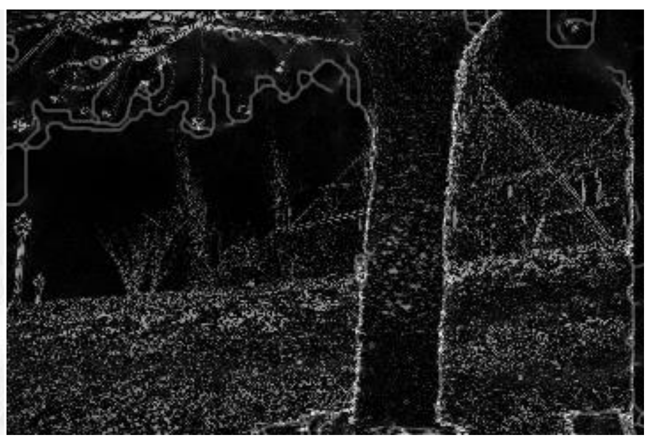
360x240 (Black)
Max speed: 7pix/frame

BA

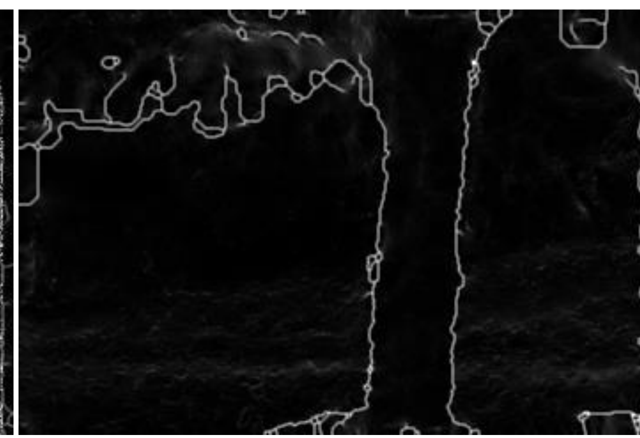


LMS

Ours



Error map



Smoothness error

Representing Moving Images with Layers

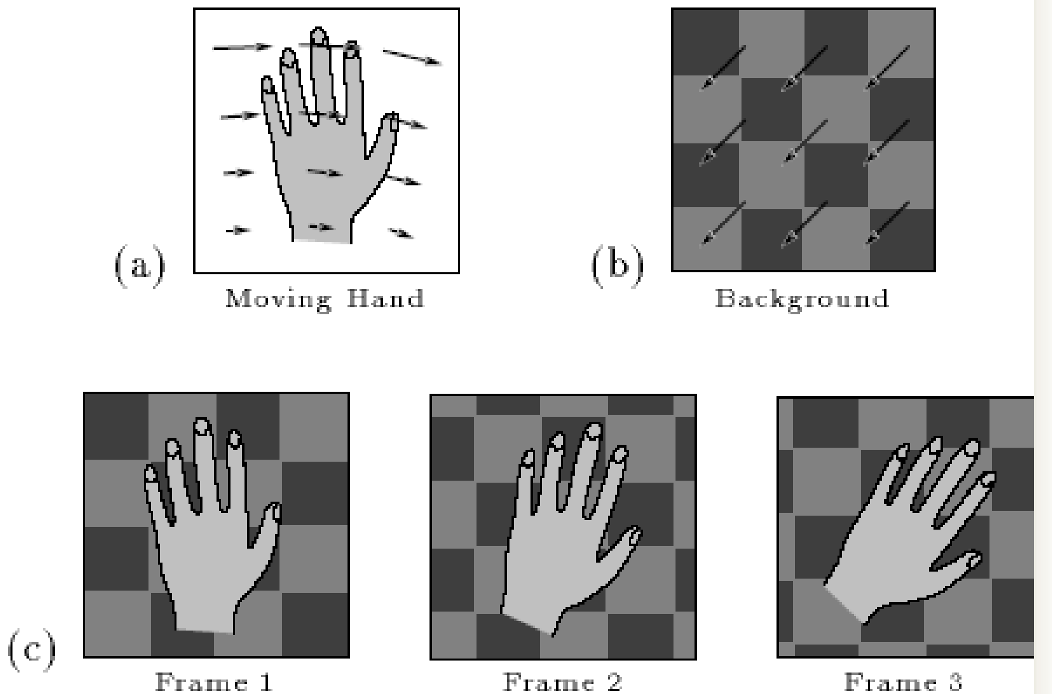
J. Y. Wang and E. H. Adelson

MIT Media Lab

Goal

- Represent moving images with sets of overlapping layers
- Layers are ordered in depth and occlude each other
- Velocity maps indicate how the layers are to be warped over time

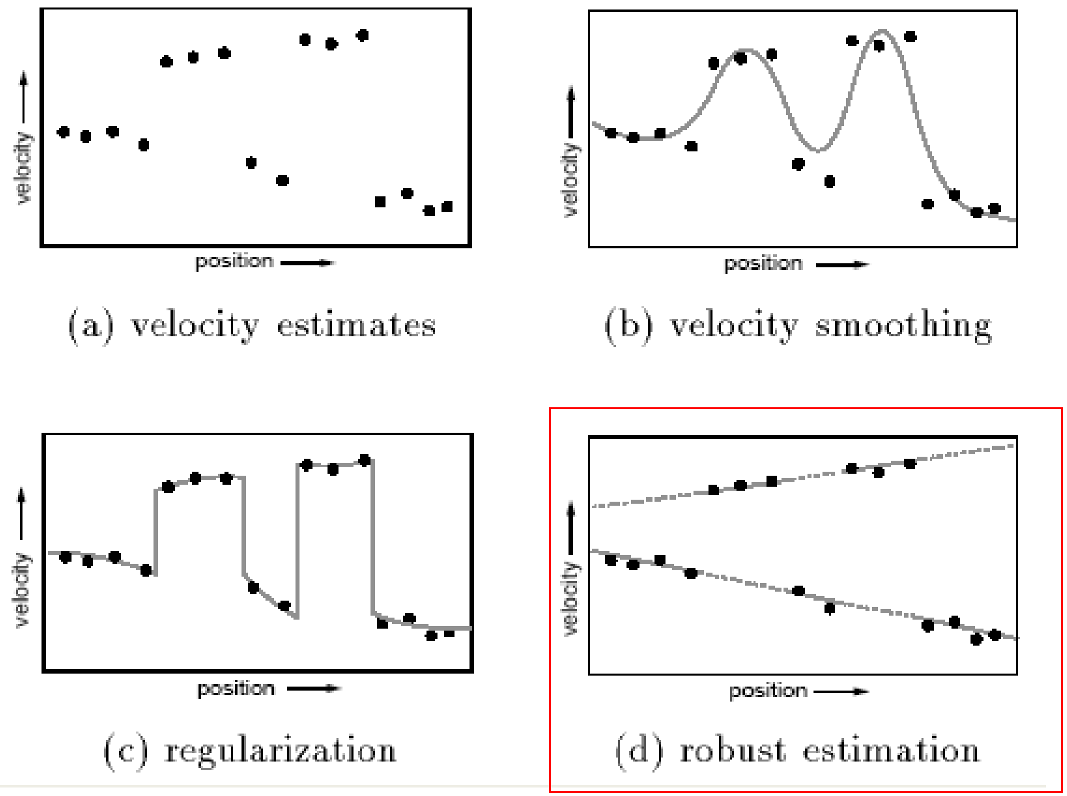
Simple Domain: Gesture Recognition



More Complex:
What are the layers?




Motion Analysis Example



(a) velocity estimates

(b) velocity smoothing

(c) regularization

(d) robust estimation

2 separate layers shown as 2 affine models (lines);

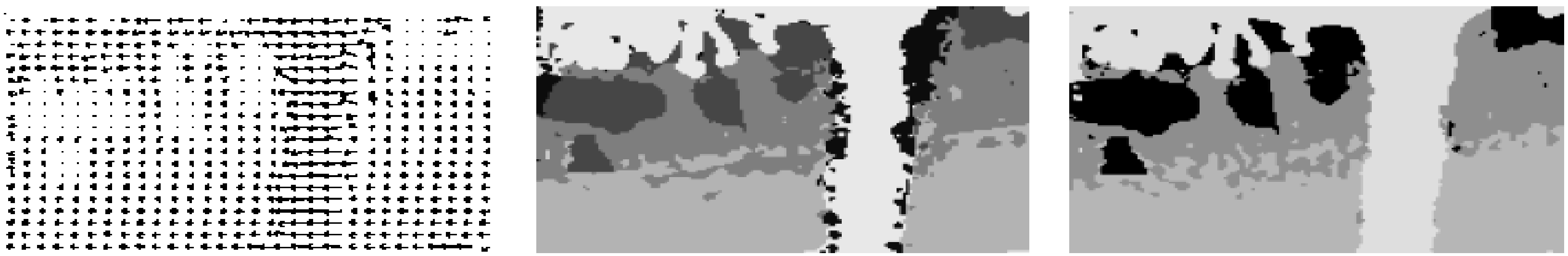
The gaps show the occlusion.

Motion Estimation Steps

1. Conventional optical flow algorithm and representation (uses multi-scale, coarse-to-fine Lucas-Kanade approach).

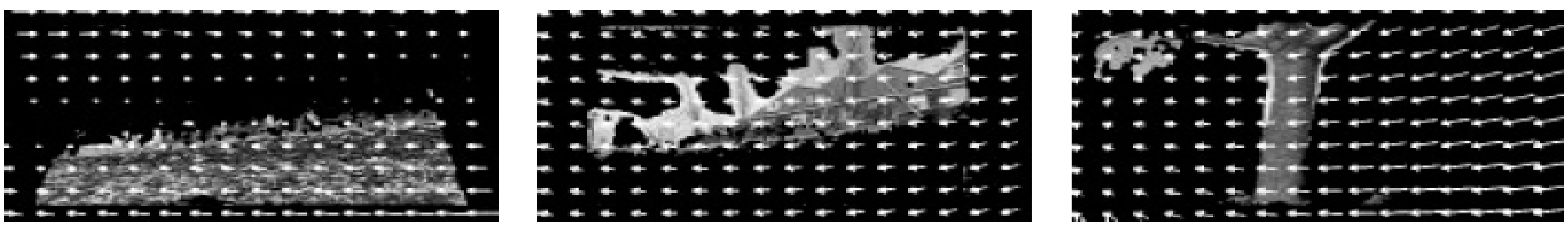
2. From the optical flow representation, determine a set of affine motions. Segment into regions with an affine motion within each region.

Results



(a) (b) (c)

Figure 11: (a) The optic flow from multi-scale gradient method. (b) Segmentation obtained by clustering optic flow into affine motion regions. (c) Segmentation from consistency checking by image warping. Representing moving images with layers.



(a) (b) (c)

Figure 12: The layers corresponding to the tree, the flower bed, and the house shown in figures (a-c), respectively. The affine flow field for each layer is superimposed.

Results

(a) (b) (c)

Figure 13: Frames 0, 15, and 30 as reconstructed from the layered representation shown in figures (a-c), respectively.

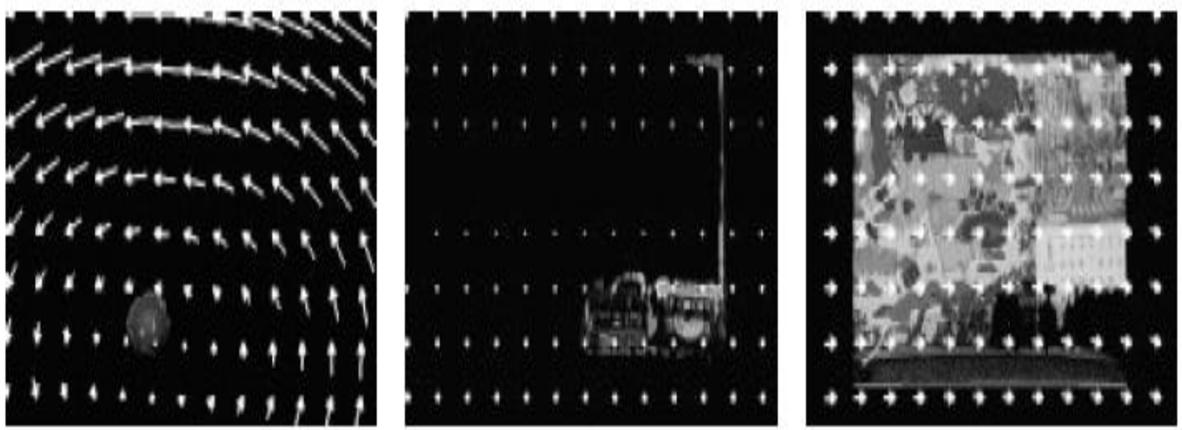
(a) (b) (c)

Figure 14: The sequence reconstructed without the tree layer shown in figures (a-c), respectively.

Results

(a) (b) (c)

Figure 15: Frames 0, 15 and 30, of MPEG Calendar sequence shown in figures (a-c), respectively.



(a) (b) (c)

Figure 16: The layers corresponding to the ball, the train, and the background shown in figures (a-c), respectively.

Summary

- Major contributions from Lucas, Tomasi, Kanade
  - Tracking feature points
  - Optical flow
  - Stereo
  - Structure from motion

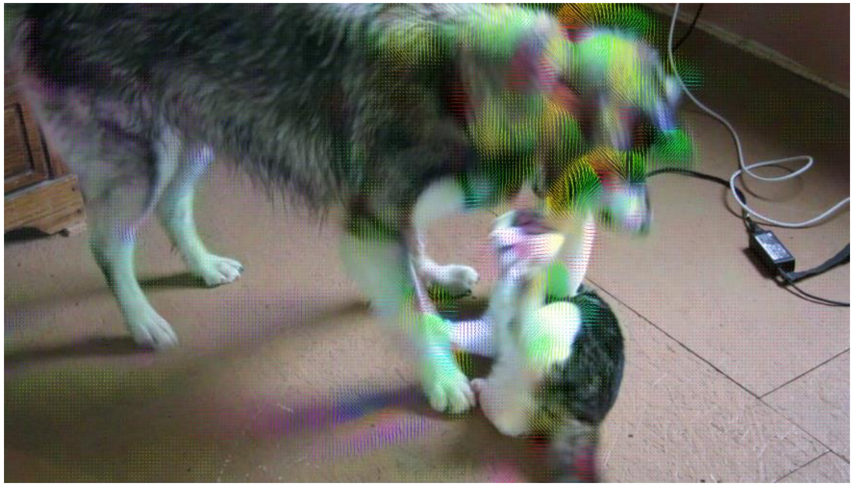
- Key ideas
  - By assuming brightness constancy, truncated Taylor expansion leads to simple and fast patch matching across frames
  - Coarse-to-fine registration
  - Global approach by former EE student Ming Ye
  - Motion layers methodology by Wang and Adelson

Back to the Homework

- For HW 4, you will implement optical flow!
- In particular, you will implement the Lucas-Kanade optical flow finder to find the optical flow between two image frames.
- Shima’s slides will give the exact details.

Homework 4
Optical Flow

Motion



Overall idea

- We'll use Lucas-Kanade's equation to find the optical flow.

- We'll need spatial and temporal gradient information for the flow equations.

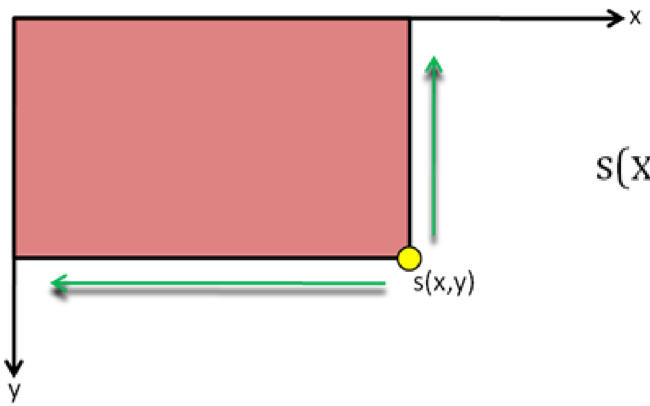
- We'll be calculating structure matrices again, so we need to do aggregated sums over regions of the image.
  - Optical flow has to run on video, so it needs to be fast! we'll use integral images to simulate smoothing with a box filter instead of smoothing with a Gaussian filter.

- We'll calculate velocity from spatial and temporal gradient information and use that to draw the motion lines.

1. Integral Image

- The Integral Image (or Summed Area Table) is used as a quick and effective way of calculating the sum of values (pixel values) or calculating the average intensity in a given image.

- When creating an Integral Image, if we go to any point (x,y), the corresponding Integral Image value is the sum of all the pixel values **above**, to the **left** and of course including the **original** pixel value of (x,y) itself.

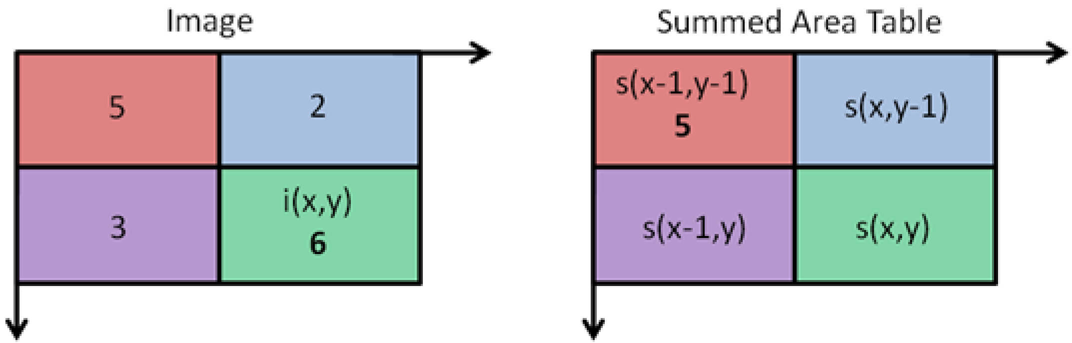


$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)$$

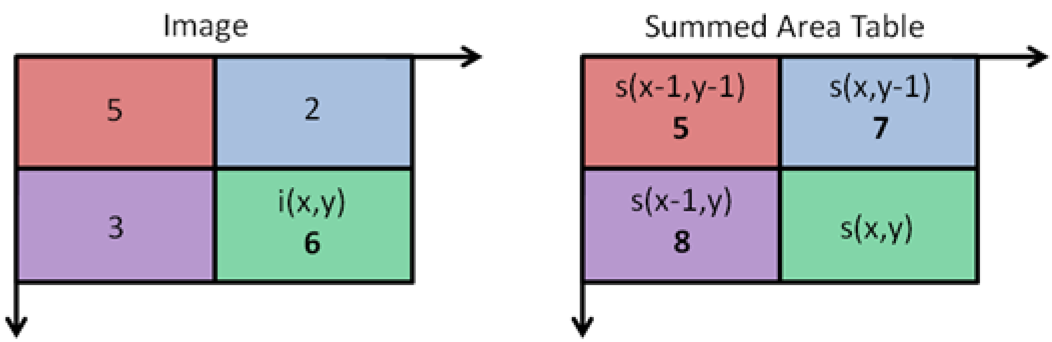
https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)$$

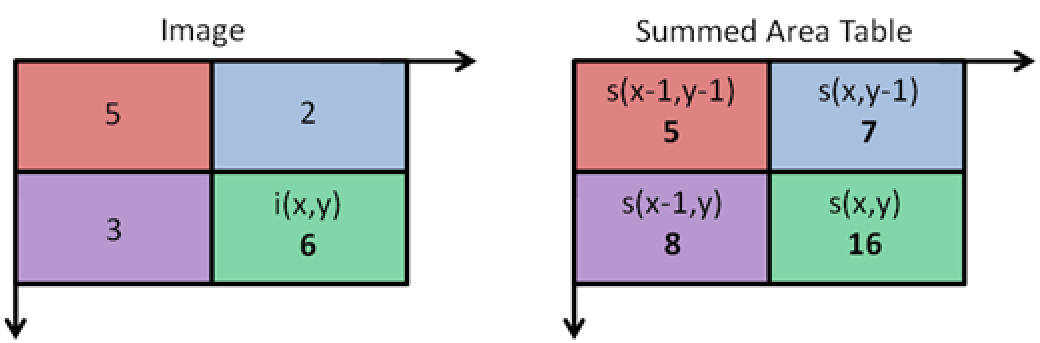
Step 1



Step 2

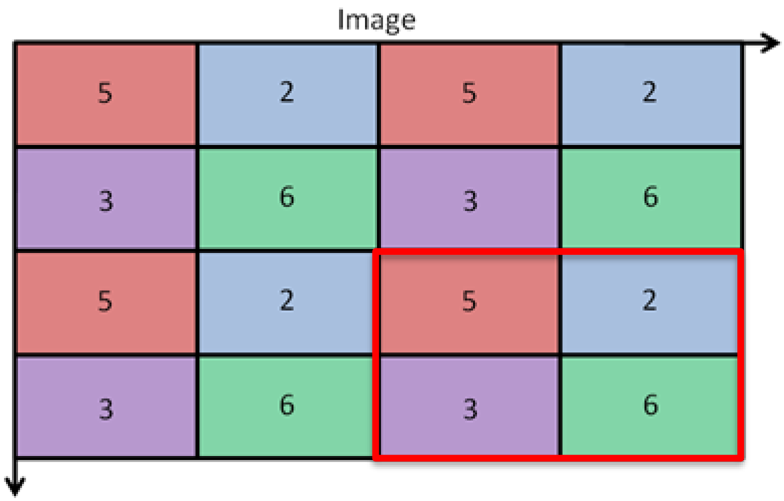


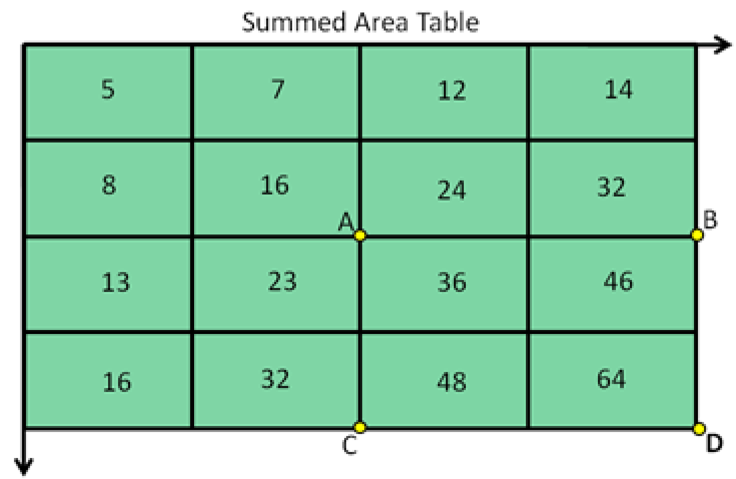
Step 3



Calculate average intensity

How to calculate area in original image, using the corresponding integral image:





Original:

Area = 5 + 2 + 3 + 6 = 16

Integral:

Area (in original image)

= [S(D) – S(C)] – [S(B) – S(A)]

= (64 – 32) – (32 – 16) = 16

Calculate average intensity

Original Image

5	4	3	8	3
3	9	1	2	6
9	6	0	5	7
7	3	6	5	9
1	2	2	8	3

Integral Image

5	9	12	20	23
8	21	25	35	44
17	36	40	55	71
24	46	56	76	101
25	49	61	89	117

Total of **9** operations.

- 9 + 1 + 2 + 6 + 0 + 5 + 3 + 6 + 5 = 37
- $\frac{37}{9} = 4.11$

Total of **4** operations.

- (76 - 20) - (24 - 5) = 37
- $\frac{37}{9} = 4.11$

TODO #1: Integral Image

- Don’t forget to `git pull` first. There are a couple of modified images and libraries.

- Fill in `image make_integral_image(image im)`

  - This function makes an integral image or summed area table from an image.
  - image im: image to process
  - returns: image I such that $I[x, y] = \sum_{\{i \leq x, j \leq y\}} im[i, j]$

TODO #2: Smoothing using integral images

- Fill in `image box_filter_image(image im, int s)` so that every pixel in the output is the average of pixels in a given window size `s`.

- Note that you must call your `make_integral_image()` in this function.

- Be careful, this is not the your old `make_box_filter()` from your other homework. It is using the integral image, and a smooth window size.

TODO #3: Lucas-Kanade optical flow

- We'll be implementing optical flow. We'll use a structure matrix but this time with temporal information as well. The equation we'll use is:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

Velocity Structure Matrix Time Matrix

TODO #3.1: Time-structure matrix

- We'll need spatial and temporal gradient information for the flow equations.

- Calculate a time-structure matrix.
  - Spatial gradients can be calculated as normal.
  - The time gradient can be calculated as the difference between the previous image and the next image in a sequence.
    - I_t = [current image] – [previous image]

TODO #3.1: Time-structure matrix

Calculate the time-structure matrix of an image pair:

- Fill in `image time_structure_matrix(image im, image prev, int s)`.

  - image im: the input image.
  - image prev: the previous image in sequence.
  - int s: window size for smoothing.
    - `im` and `prev` to grayscale (given in the code).
    - Hint: use `sub_image` to subtract `im` and `prev`.
    - Calculate gradients and structure matrix and smooth (hint: use your gx and gy functions from HW2)

  - …next slide: return

TODO #3.1: Time-structure matrix

Calculate the time-structure matrix of an image pair:

- Fill in `image time_structure_matrix(image im, image prev, int s)`.
  - returns: structure matrix which has 5 channels:
    - 1st channel is $I_x I_x$
    - 2nd channel is $I_y I_y$
    - 3rd channel is $I_x I_y$
    - 4th channel is $I_x I_t$
    - 5th channel is $I_y I_t$
  - Each channel is a vector with the structure of an image.
  - Use `make_box_filter()` to smooth.

TODO #3.2: Calculating velocity from the time-structure matrix

Calculate the velocity given a time-structure image

- Fill in `image velocity_image(image S, int stride)`
  – Image S is the output of time_structure_matrix which you already summed and smooth.

- For each pixel, fill in the `matrix M`, invert it, and use it to calculate the velocity.

$$M = \begin{bmatrix} I_x(q_i)^2 & I_x(q_i)I_y(q_i) \\ I_y(q_i)I_x(q_i) & I_y(q_i)^2 \end{bmatrix}$$

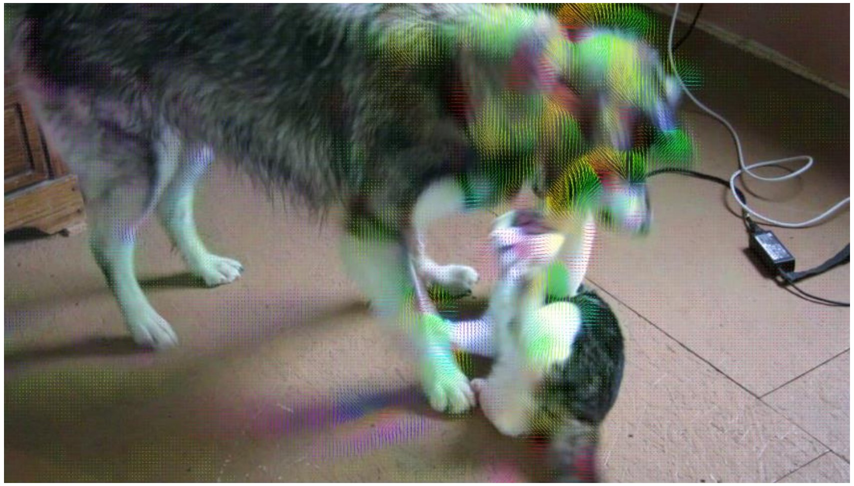
$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = -M^{-1} * \begin{pmatrix} I_{x_t} \\ I_{y_t} \end{pmatrix}$$

Draw motion with optical flow

`optical_flow_images()` will call your `time_structure_matrix()` and `velocity_image()`. Then `draw_flow()` will draw lines of motion on the image.

Try calculating the optical flow between two dog images using tryhw4.py.

```
a = load_image("data/dog_a.jpg")
b = load_image("data/dog_b.jpg")
flow = optical_flow_images(b, a, 15, 8)
draw_flow(a, flow, 8)
save_image(a, "lines")
```



Optical flow demo using OpenCV

- This part is optional and is a 1 point extra credit, but it is fun to do.
- Using OpenCV we can get images from the webcam and display the results in real-time. Try installing OpenCV and enabling OpenCV compilation in the Makefile (set `OPENCV=1` in the first line). Then uncomment this line in `tryhw4.py`:

```
optical_flow_webcam(15,4,8)
```

- Turn in your `flow_image.c` file on Canvas.

Have fun!

And stay healthy..