

Computer Vision 1

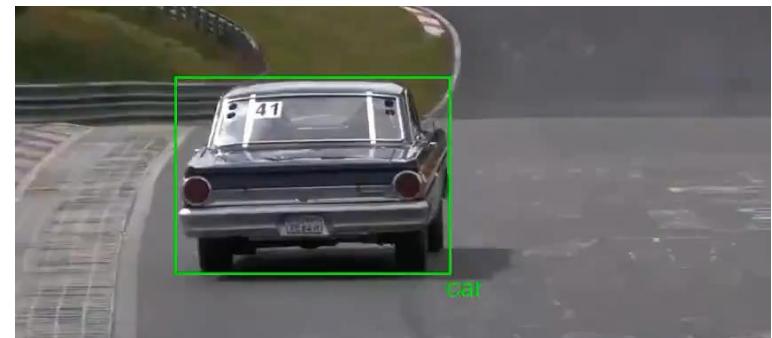
HC3b

Motion Optical Flow Tracking

Dr. Martin Oswald, Dr. Dimitris Tzionas, Dr. Arun Mukundan,
[m.r.oswald, d.tzionas, a.mukundan]@uva.nl

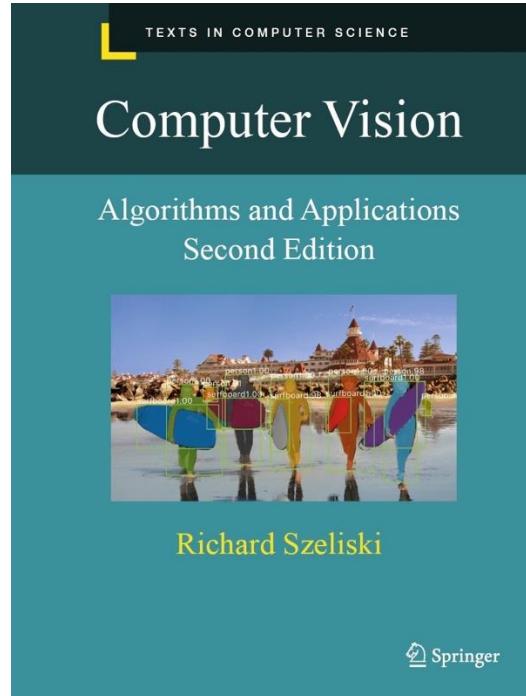
Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking



Textbook

- Sec. 9.1.1
- Sec. 9.1.3
- Sec. 9.3 (9.3.1optional)

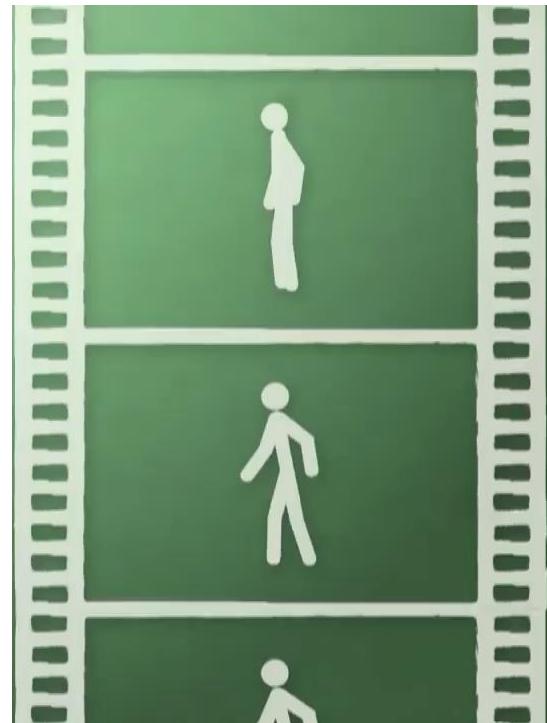
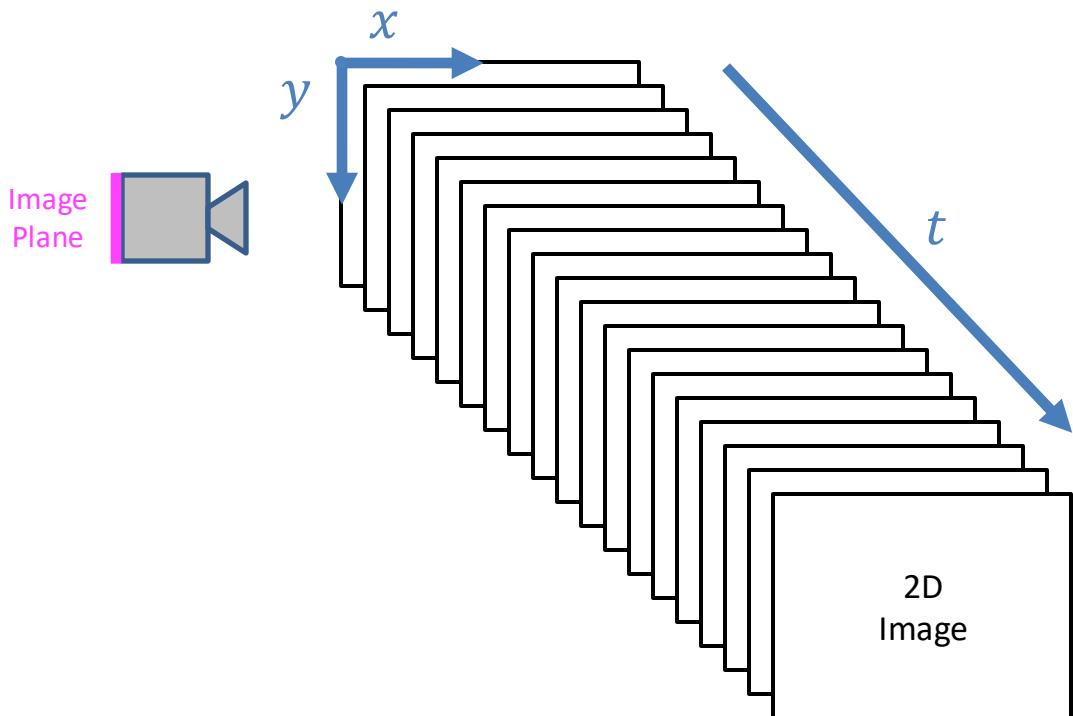


Springer

Outline

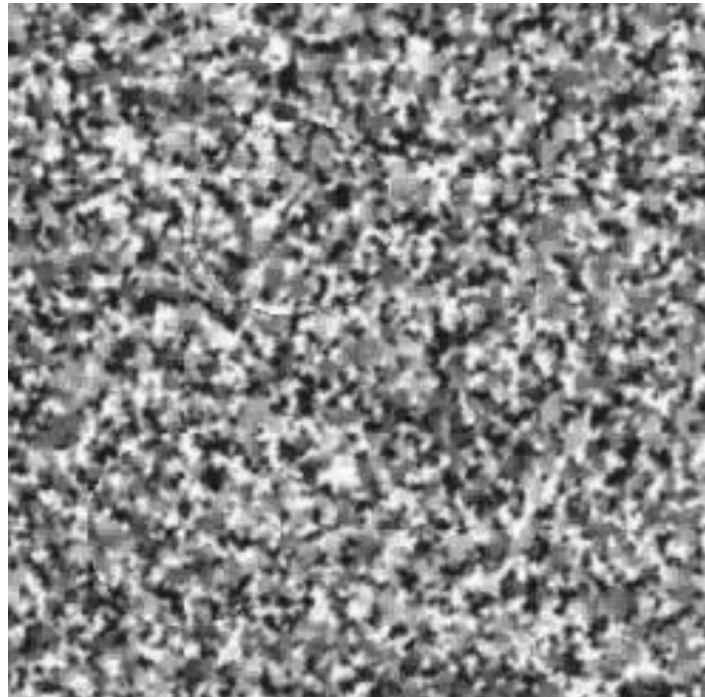
- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Motion – Video



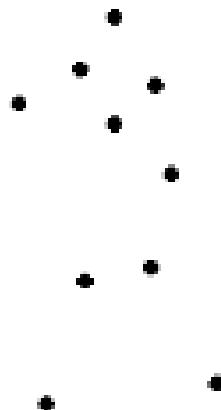
Motion Perception

- Motion is a powerful perceptual cue
- Sometimes, it is the only cue



Motion Perception

- Even ‘**impoverished**’ motion data evoke a **strong percept**



G. Johansson

'Visual Perception of Biological Motion and a Model For Its Analysis'
Perception and Psychophysics, 1973.

Making Sense of Video

An often-told story



Aspiring young
graduate student

What are the three
most important
problems in computer
vision?

Correspondence,
Correspondence,
Correspondence!



[Takeo Kanade](#)

Many Forms of Correspondence

Bounding box

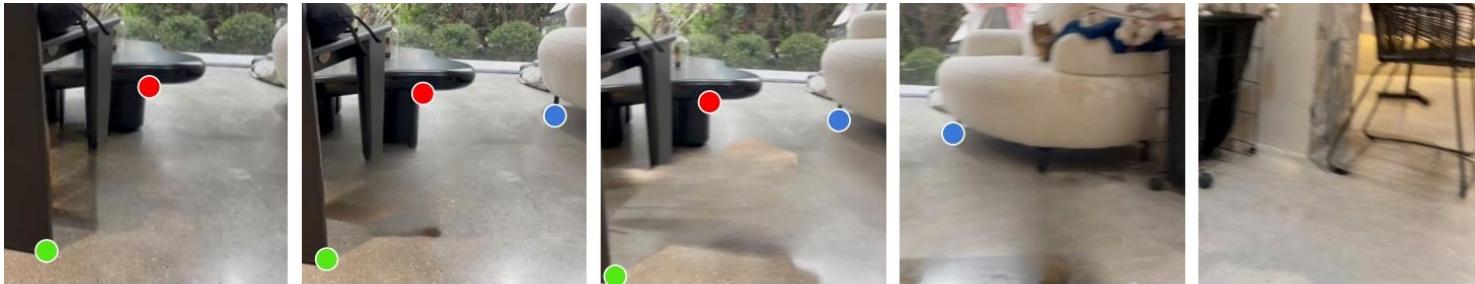
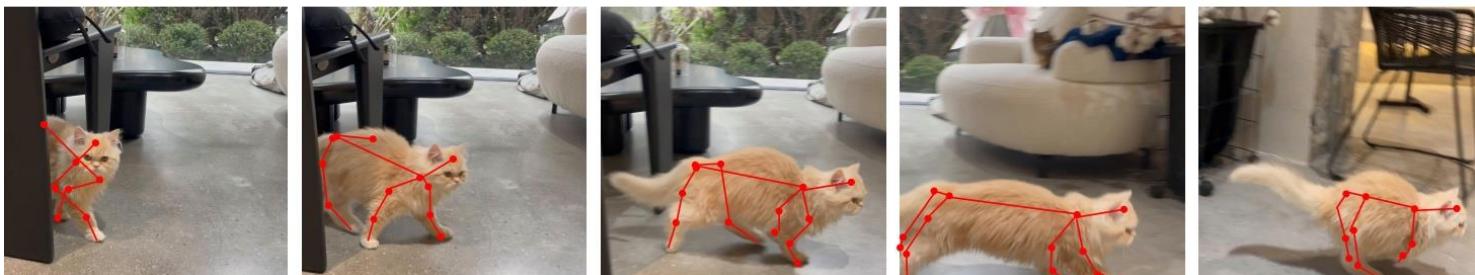


Segmentation



Many Forms of Correspondence

Keypoints

Pose/semantic
keypoints

Dense Correspondences

Optical Flow (today's topic)



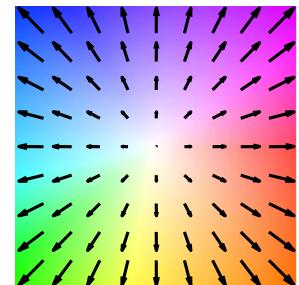
Input

[Liu *et al.* CVPR'08]



Optical Flow

(2D motion vector)



Color key

[Baker *et al.* IJCV'11]

Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Motion Perception

Animal moves

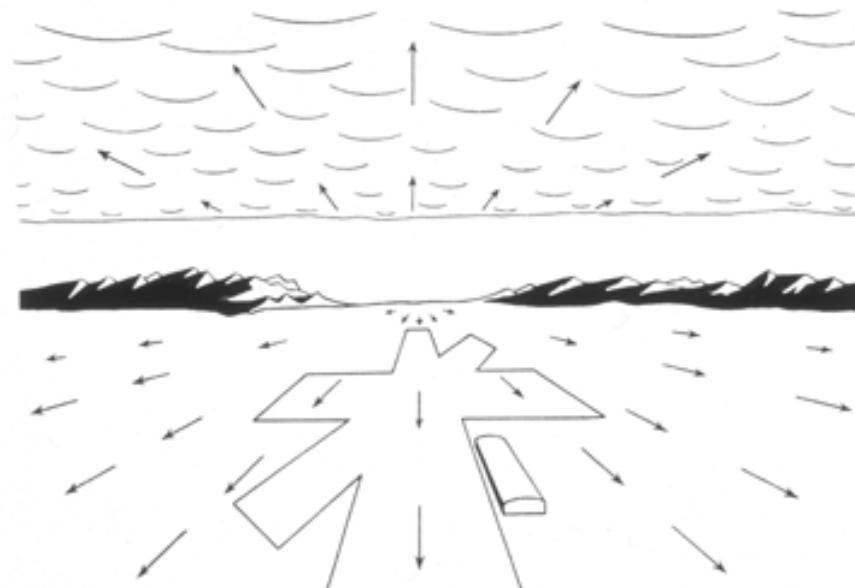


Changing luminance pattern

on 'optic array' [Gibson]
or camera (img plane)

Optical Flow:

'Motion' of this pattern
('apparent motion')



Assumption:

Gives *direct access* to properties of the world that are important for organism to *survive*.

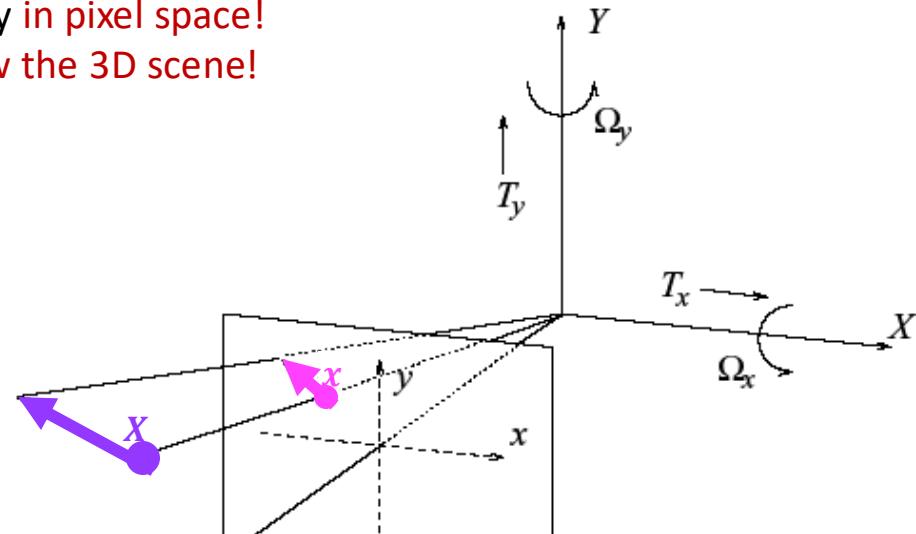
- Rabbit runs to hide in hole
- Lion jumps to eat you

Motion Field (MF) & Optical Flow (OF)

2D Motion Field (MF): Describes *projection* of the **3D motion** of **3D scene points** onto **2D image plane**
Note: Knows the 3D scene!

2D Optical-Flow Field (OF): Describes *apparent motion* in *images*
Note: Reasoning only in **pixel space!**
Does not know the 3D scene!

$$\begin{aligned}x &= (x, y) \\X &= (X, Y, Z)\end{aligned}$$



MF & OF – Thought Experiment #1

2D Motion Field (MF):

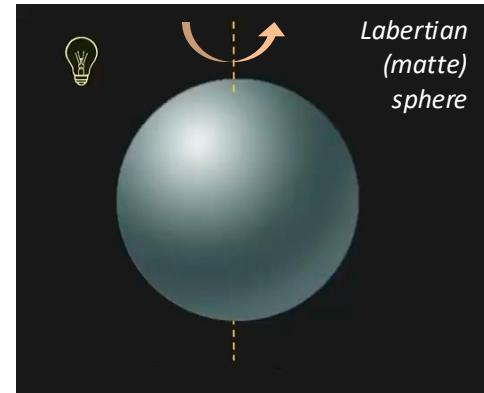
Describes *projection* of the **3D motion** of **3D scene points** onto **2D image plane**

Note: Knows the 3D scene!

2D Optical-Flow Field (OF): Describes *apparent motion* in *images*

Note: Reasoning only in **pixel space!**

Does not know the 3D scene!

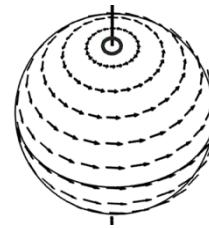


Example: Rotating **Labertian** ('matte') sphere

Motion Field → We know how the **3D sphere points rotate** ('3D Scene Flow')

Project the **3D motion vectors** onto **2D image plane** & get:

MF == Non-zero 2D vectors w rough direction **left→right**

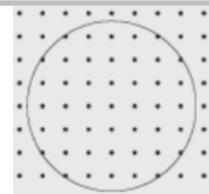


https://en.wikipedia.org/wiki/Optical_flow

Optical Flow → We do **not** know how the **3D sphere rotates** ('3D Scene Flow')

We **just** judge by **looking at pixels**

OF == Constant (no apparent motion) == Zero-length 2D vectors



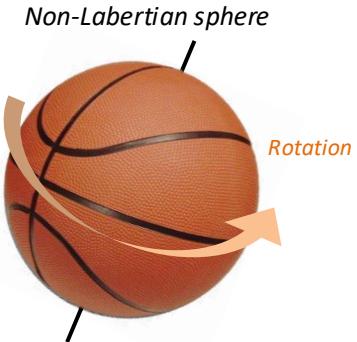
<http://www.cs.toronto.edu/~jepson/csc2503/opticalFlow.pdf>

MF & OF – Thought Experiment #2

2D Motion Field (MF):

Describes *projection* of the **3D motion** of **3D scene points** onto **2D image plane**
 Note: Knows the 3D scene!

2D Optical-Flow Field (OF): Describes *apparent motion* in *images*
 Note: Reasoning only in **pixel space!**
 Does not know the 3D scene!

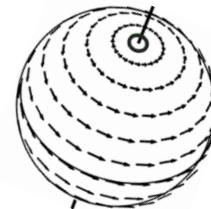


Example: Rotating **Non-Labertian** sphere

Motion Field → We know how the **3D sphere points rotate** ('3D Scene Flow')

Project the **3D motion vectors** onto **2D image plane** & get:

MF == Non-zero 2D vectors w rough direction **left→right**

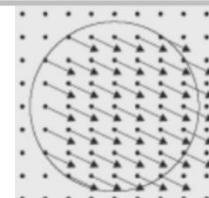


https://en.wikipedia.org/wiki/Optical_flow

Optical Flow → We do **not** know how the **3D sphere rotates** (3D Scene Flow)

We **just** judge by **looking at pixels**

OF == Non-zero 2D vectors w rough direction **left→right**



<http://www.cs.toronto.edu/~jepson/csc2503/opticalFlow.pdf>

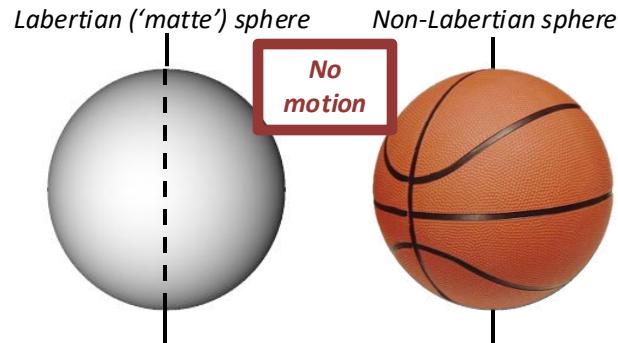
MF & OF – Thought Experiment #3

2D Motion Field (MF):

Describes *projection* of the **3D motion** of **3D scene points** onto **2D image plane**
 Note: Knows the 3D scene!

2D Optical-Flow Field (OF):

Describes *apparent motion* in *images*
 Note: Reasoning only in **pixel space!**
 Does not know the 3D scene!

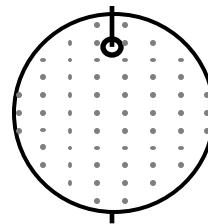


Example: No sphere rotation

Motion Field → We know that the **3D sphere points don't rotate**

Project zero-length **3D motion vectors** on 2D and get...

MF == Constant motion == Zero-length 2D vectors

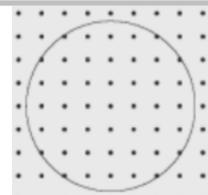


https://en.wikipedia.org/wiki/Optical_flow

Optical Flow → We do **not** know how the 3D sphere rotates (3D Scene Flow)

We **just** judge by **looking at pixels**

OF == Constant (no apparent motion) == Zero-length 2D vectors



<http://www.cs.toronto.edu/~jepson/csc2503/opticalFlow.pdf>

MF & OF – Thought Experiment #4

2D Motion Field (MF):

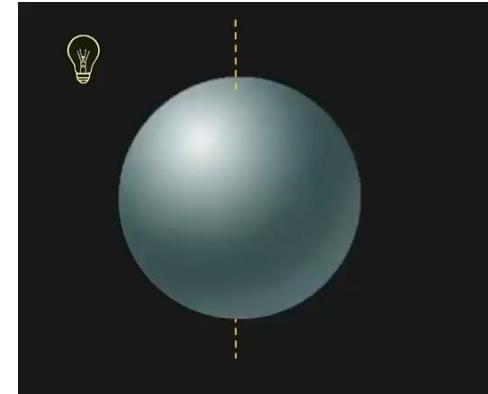
Describes *projection* of the **3D motion** of **3D scene points** onto **2D image plane**

Note: Knows the 3D scene!

2D Optical-Flow Field (OF):

Describes *apparent motion* in *images*

Note: Reasoning only in **pixel space!**
Does not know the 3D scene!

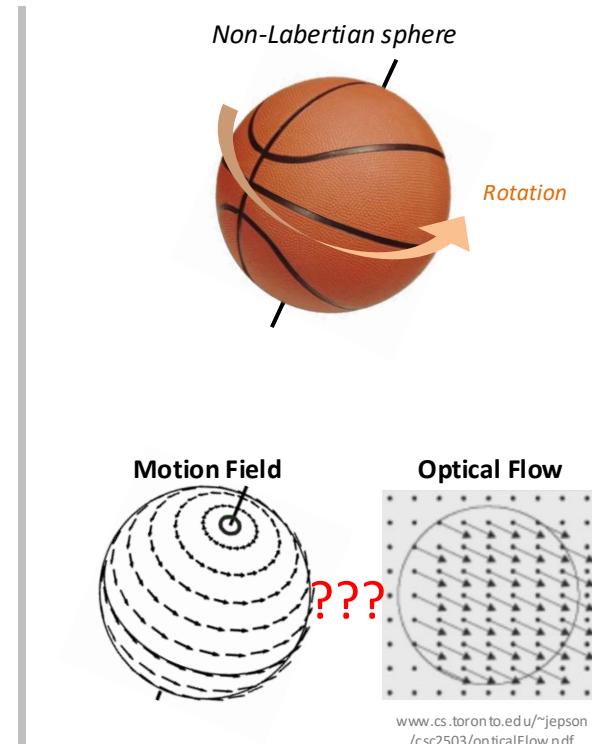
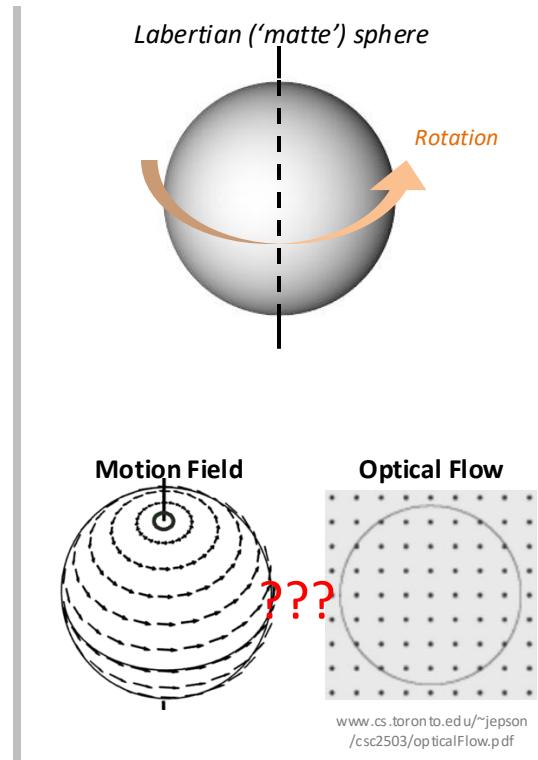
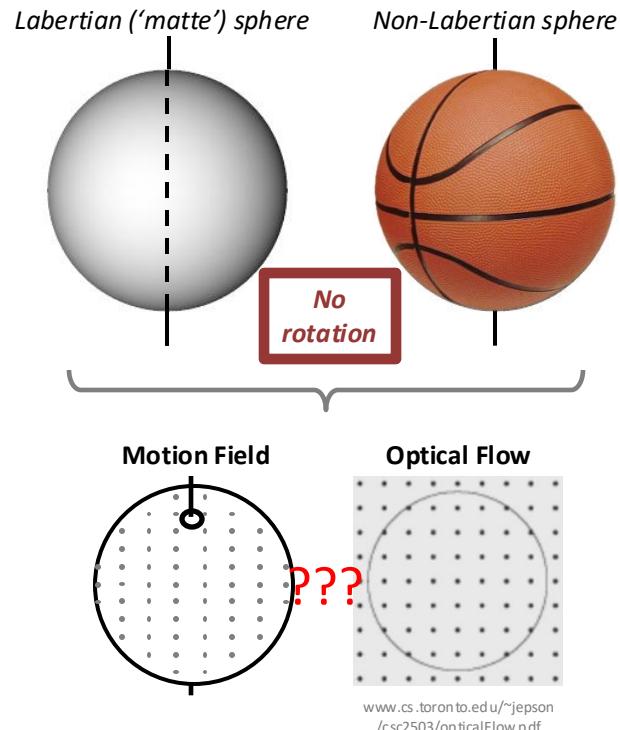


Example: No sphere rotation – But light moves!

MF == Constant motion
== Zero-length 2D vectors

OF == Non-zero 2D vectors
direction depends on **apparent motion**
due to light motion

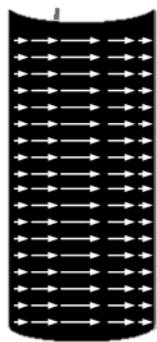
MF & OF – Thought Experiments – Summary



Optical Flow != Motion Field



Barber
Pole



Motion
Field

(3D aware)



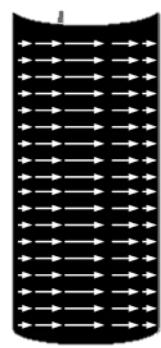
Optical
Flow

(only 2D aware)

Optical Flow != Motion Field



'Peephole'
(explains Optical Flow)



Motion
Field
(3D aware)



Optical
Flow
(only 2D aware)

Optical Flow

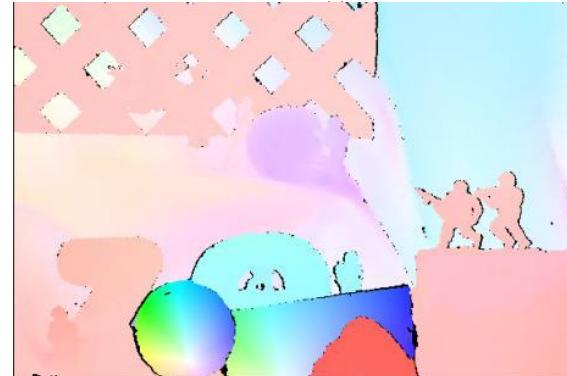
Apparent Motion (displacement) of Pixels



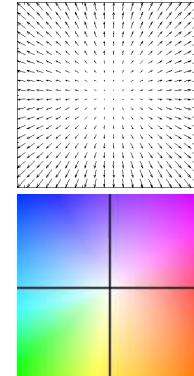
<https://vision.middlebury.edu/flow/data>

<https://eval-color-allframes.zip>

Seq. 'Army' , frames: 11 & 12



Optical Flow
(apparent displacement)



Color-coding

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

Image intensity (pixel value) at time t & location $x = (x, y)$

$$u(x, y)$$

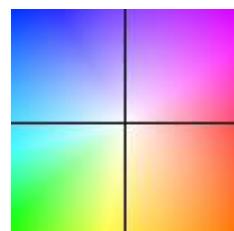
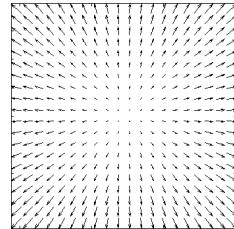
$$v(x, y)$$

Horizontal component

Vertical component

Color → Direction
Saturation → Magnitude

Optical Flow – Goal



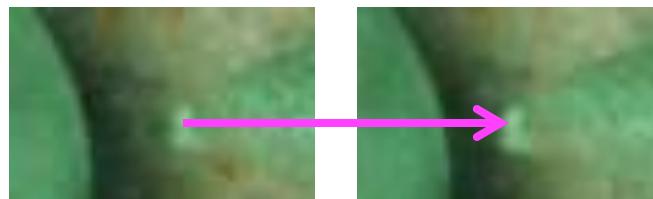
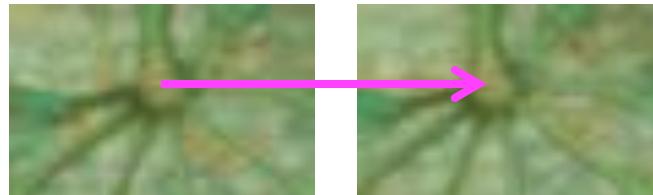
The ***apparent*** 2D image ***motion of pixels***
from one image frame to the next one
in a video sequence

OF – Assumption #1 – Brightness Constancy

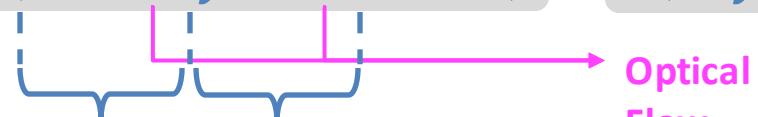
CV

The most foundational
assumption for OF

Assumption #1: Brightness Constancy



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



Displacement @ next frame

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

Alternative notation

OF – Assumption #2 – Small Motion

- Estimate OF $u, v \rightarrow$ Take these out of function I
- Small motion == Small Flow
- Image: differentiable function

Use Taylor's expansion \rightarrow

* To simplify our notation

In theory: dt is infinitesimally small

In practice: $dt = \frac{1}{FPS}$ of camera

Estimating
2D Optical Flow
for 1 point
is ill posed!

Assumption #2: Small motions

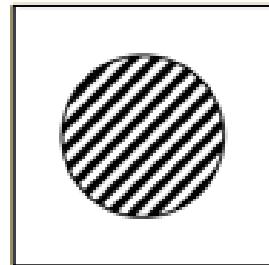
$$\begin{aligned}
 I(x + u, y + v, t + 1) &= I(x, y, t) \\
 \text{---} I(x, y, t) + dx \frac{\partial}{\partial x} I(x, y, t) + dy \frac{\partial}{\partial y} I(x, y, t) + dt \frac{\partial}{\partial t} I(x, y, t) &= I(x, y, t) \\
 u \frac{\partial}{\partial x} I(x, y, t) + v \frac{\partial}{\partial y} I(x, y, t) + \frac{\partial}{\partial t} I(x, y, t) &= 0 \\
 u I_x + v I_y + I_t &= 0
 \end{aligned}$$

Can be approx. computed
2 unknowns in 1 equation

Outline

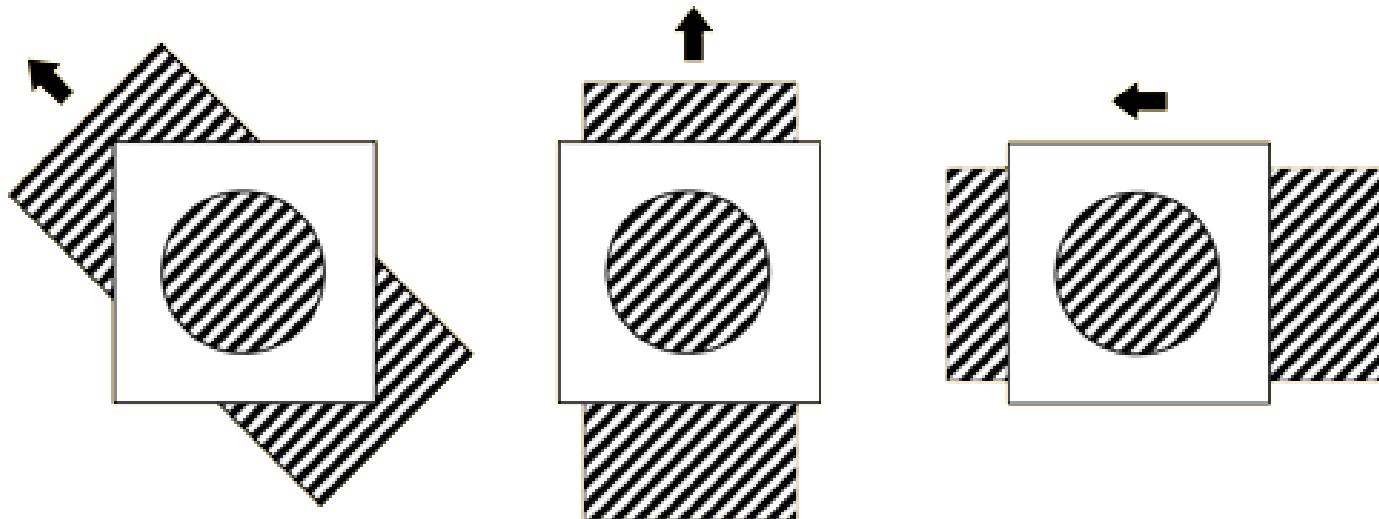
- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Optical Flow – Aperture Problem



What do we
see through
the hole?
("point")

Optical Flow – Aperture Problem



<https://cdn.sinauer.com/wolfe4e/wa08.02.html>

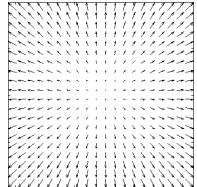
Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

OF - Assumption #3: Spatial Smoothness

For method of
Lucas-Kanade

Assumption #3: Spatial Smoothness



Neighboring pixels in the image are likely to belong to the same surface

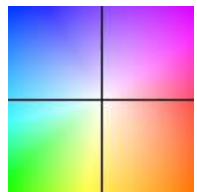
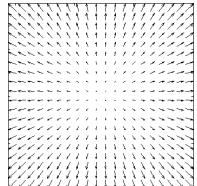
Surfaces are mostly smooth

Neighboring pixels will have similar flow

OF – Assumption #3: Spatial Smoothness

For method of
Lucas-Kanade

Assumption #3: Spatial Smoothness

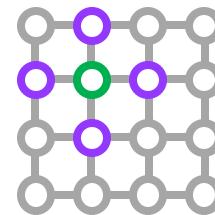


Neighboring points
have the **same**
optical flow

i.e., the spatial derivative
of optical flow is zero

$$\begin{cases} u_{\mathbf{x}} = u_{x_i} \\ v_{\mathbf{x}} = v_{x_i} \end{cases}$$

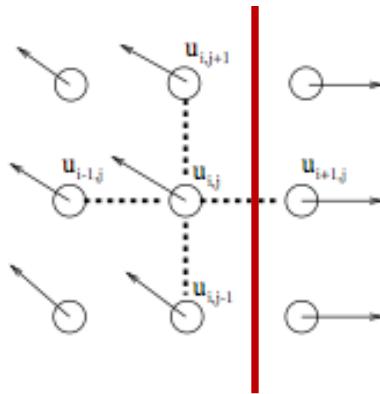
$x_i \in G(\mathbf{x})$
Graph
neighborhood
of 'central' point
 $\mathbf{x} = (x, y)$



OF – Assumption #3: Spatial Smoothness

For method of
Lucas-Kanade

Assumption #3: Spatial Smoothness



Smoothness: Neighboring pixels *usually* belong to *same* surface

Except when they *don't* → Different objects separated

by **motion boundaries/discontinuities**

Optical Flow – Lucas-Kanade Method

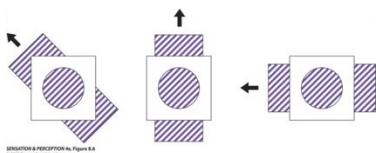
For 1 point:

$$uI_x + vI_y + I_t = 0$$

2 unknowns
1 equation

Problem

Estimating 2D Optical Flow
for 1 point is ill posed ...
(aperture problem)



SENSATION & PERCEPTION 4e, Figure 8.6

Solution

so, let's consider **more points**
 $n \geq 2$ and assume that the...

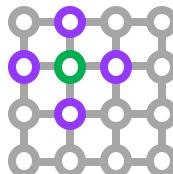
- central point $\mathbf{x} = (x, y)$ and
- surrounding points $\mathbf{x}_i = (x_i, y_i)$

have **same Optical Flow** $\mathbf{u} = (u, v)$

$$\mathbf{u}_x = \mathbf{u}_{x_i}$$

$$\mathbf{x}_i \in G(\mathbf{x})$$

$$i = 1, \dots, n$$



For all $i = 1, \dots, n$:

$$uI_{x_i} + vI_{y_i} = -I_t$$

$$uI_x(\mathbf{x}_i, t) + vI_y(\mathbf{x}_i, t) = -I_t(\mathbf{x}_i, t)$$

Stacking equations for $i = 1, \dots, n$:

$$\begin{bmatrix} I_x(\mathbf{x}_1, t) & I_y(\mathbf{x}_1, t) \\ I_x(\mathbf{x}_2, t) & I_y(\mathbf{x}_2, t) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n, t) & I_y(\mathbf{x}_n, t) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{x}_1, t) \\ I_t(\mathbf{x}_2, t) \\ \vdots \\ I_t(\mathbf{x}_n, t) \end{bmatrix}$$

$$\mathbf{A}\mathbf{u} = -\mathbf{b}$$

Least-squares solution

$$\mathbf{u} = -(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

Structure tensor

$$\mathbf{A}^T \mathbf{A}$$

$$\left[\begin{array}{cc} \sum_{i=1}^n I_x^2(\mathbf{x}_i, t) & \sum_{i=1}^n I_x(\mathbf{x}_i, t)I_y(\mathbf{x}_i, t) \\ \sum_{i=1}^n I_x(\mathbf{x}_i, t)I_y(\mathbf{x}_i, t) & \sum_{i=1}^n I_y^2(\mathbf{x}_i, t) \end{array} \right]$$

Depend on the image neighborhood we focus on

Consider:

- Which neighborhood to choose
- Whether \mathbf{x}_i should be weighted in least-squares error
- Works **only** for **small displacements** (eg max ~2 pixels)
- For **big displacements** ... ??? (see next slide)

Outline

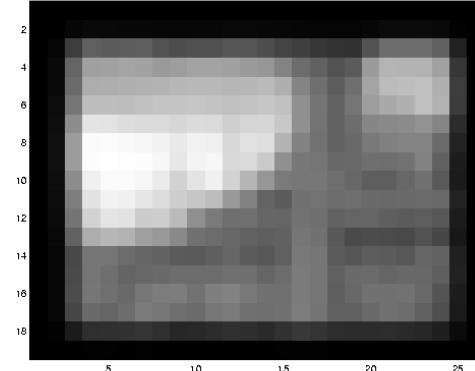
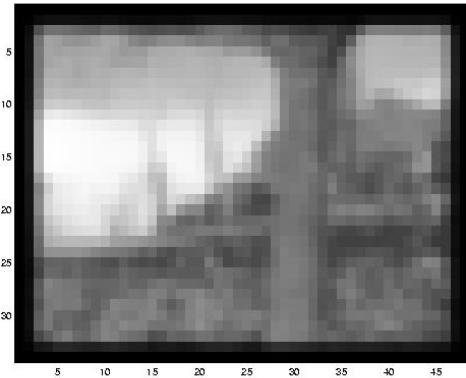
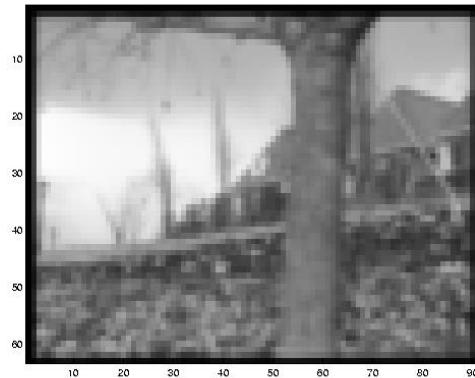
- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Revisiting Small Motion

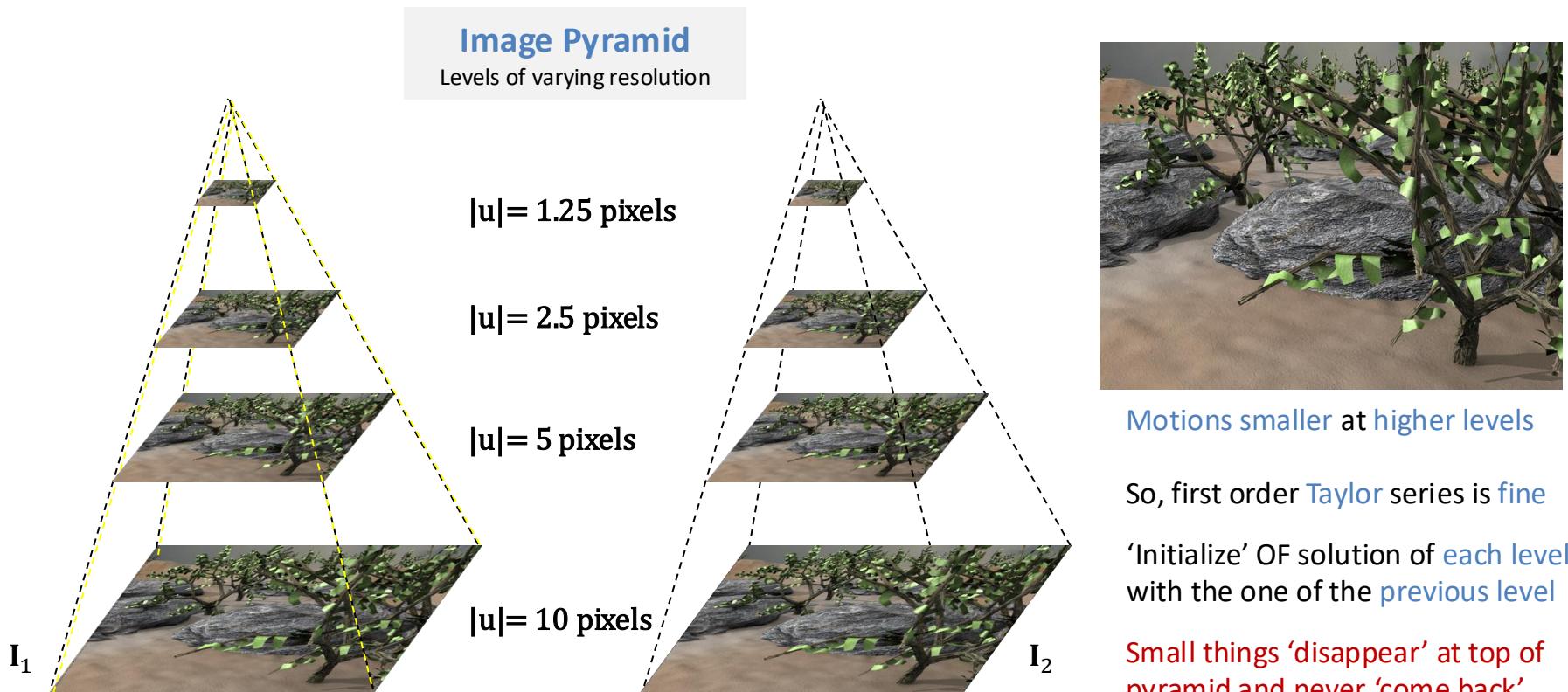
- Is this motion **small enough?**
 - Probably not—it's **much larger than 1 pixel**
 - How might we solve this problem?



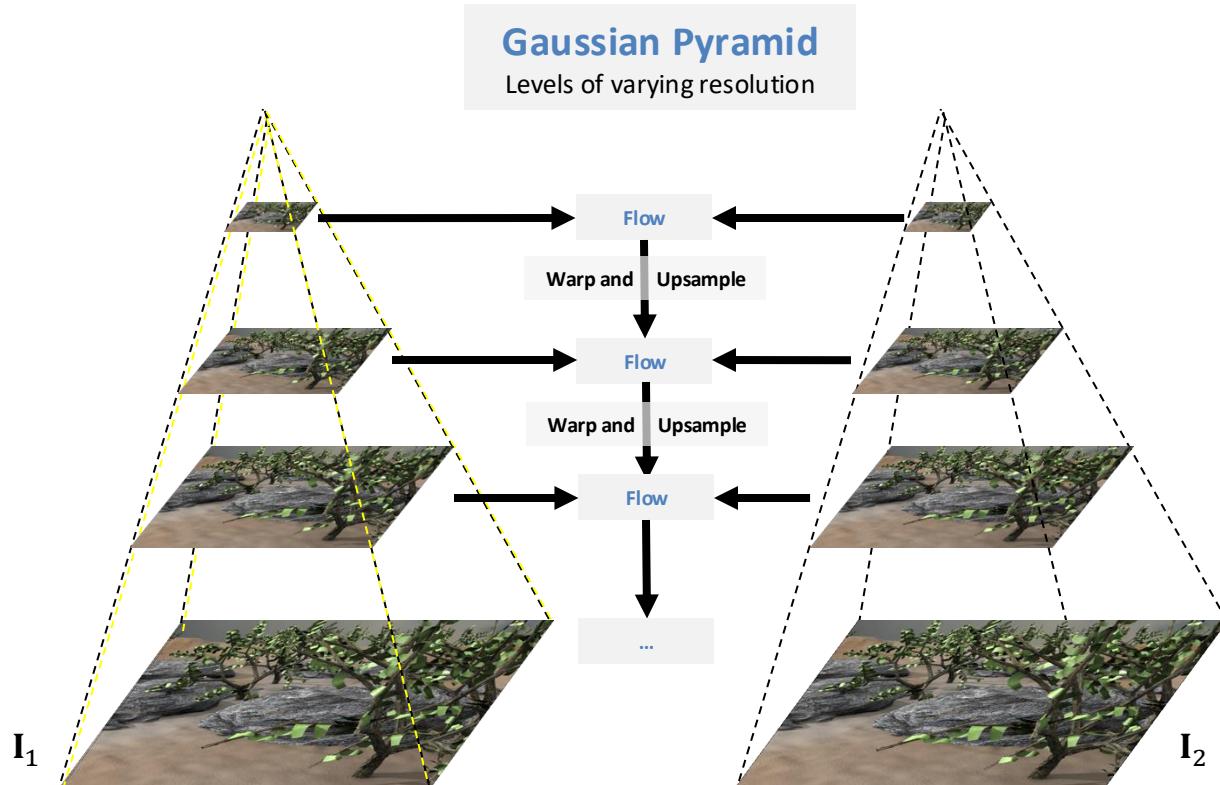
Reduce the resolution!



Optical Flow – Coarse-to-fine Estimation



Optical Flow – Coarse-to-fine Estimation



Motions smaller at higher levels

So, first order Taylor series is fine

'Initialize' OF solution of each level with the one of the previous level

Small things 'disappear' at top of pyramid and never 'come back'

Optical Flow – Coarse-to-fine Estimation

Warping Operation



1st frame and 2nd frame

Big difference



1st frame and warped 2nd frame

Smaller difference!

Optical Flow – Coarse-to-fine Estimation

CVN



Original Pair



Flow Aligned Pair

Optical Flow – Coarse-to-fine Estimation

CVN



Original Pair



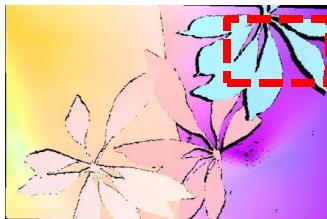
Flow Aligned Pair

Optical Flow – Challenges

Baker et al. 2007



Ground truth



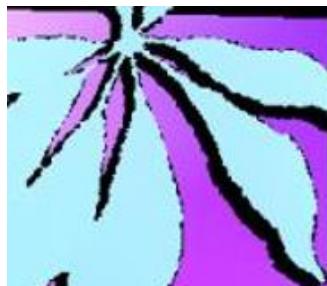
Horn & Schunck, 1981 *



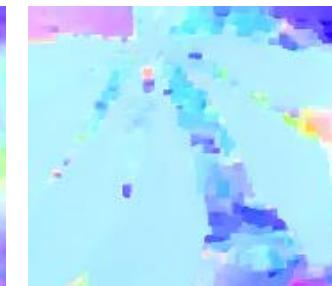
Black & Anandan, 1996 *



* Some
'classic'
methods



Occluded regions
in black



- Fine Structures
- Motion boundaries
- Occlusions

Challenging for
Optical Flow
estimation

Optical Flow – Challenges



Large motion, motion blur



Texture-less regions



Occlusions

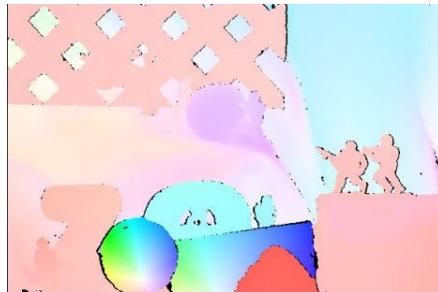


Lighting changes, noise ...

Optical Flow – Two Classic Methods



Color Images



Ground truth



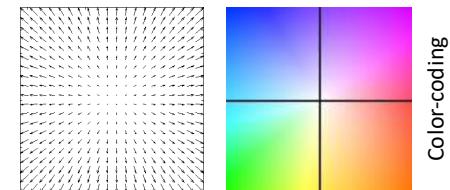
Horn & Schunck, 1981 (HS) *



Black & Anandan, 1996 (BA) *

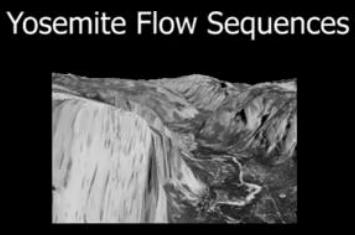
<https://vision.middlebury.edu/flow/data>
Seq. 'Army'

* Some 'classic' methods
(pre-Deep Learning)



Optical Flow – Datasets

Small-scale Datasets (Evaluation)



Performance of optical flow techniques
John L. Barron, David J. Fleet, Steven S. Beauchemin
IJCV 1994



A Database and Evaluation Methodology for Optical Flow
Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth,
Michael J. Black, Richard Szeliski
IJCV 2011

Large-scale Datasets (Evaluation + Machine Learning)



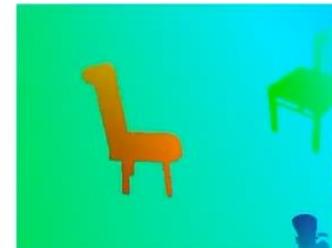
‘Sintel’ dataset

A naturalistic open source movie for optical flow evaluation
Daniel J. Butler, Jonas Wulff,
Garrett B. Stanley, Michael J. Black
ECCV 2012
<http://sintel.is.tue.mpg.de>



‘Flying Chairs’ dataset

FlowNet: Learning optical flow with convolutional networks
Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg,
Philip Hausser, Caner Hazirbas, Vladimir
Golkov, Patrick Van Der Smagt, Daniel
Cremers, Thomas Brox
ICCV 2015
[https://lmb.informatik.uni-freiburg.de/
resources/datasets/
FlyingChairs.en.html](https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html)



Sintel - Open Movie by
Blender Foundation

<https://youtu.be/eRsGyueVLvQ>

Outline

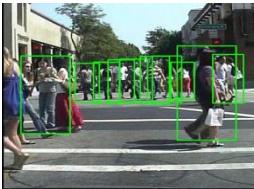
- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Optical Flow – Applications

Image processing



[Liu & Freeman 2010]



[Walk et al. 2010]

- Video denoising
- Video super resolution
- Video compression
- Pedestrian detection

...

Computer Graphics



[Hays & Essa 2004]



[Wang et al. 2010]

- Painterly animation
- Motion magnification
- Video retargeting

...

Games

Plant growth

Biology: cell motion,

heart wall motion, ...

Ocean currents, Meteorology, ...

Particle image velocimetry, ...

...

Optical Flow – Applications

Video Super-resolution

[Liu & Sun CVPR 2011, TPAMI 2014]



Optical Flow – Applications

Video Stabilization



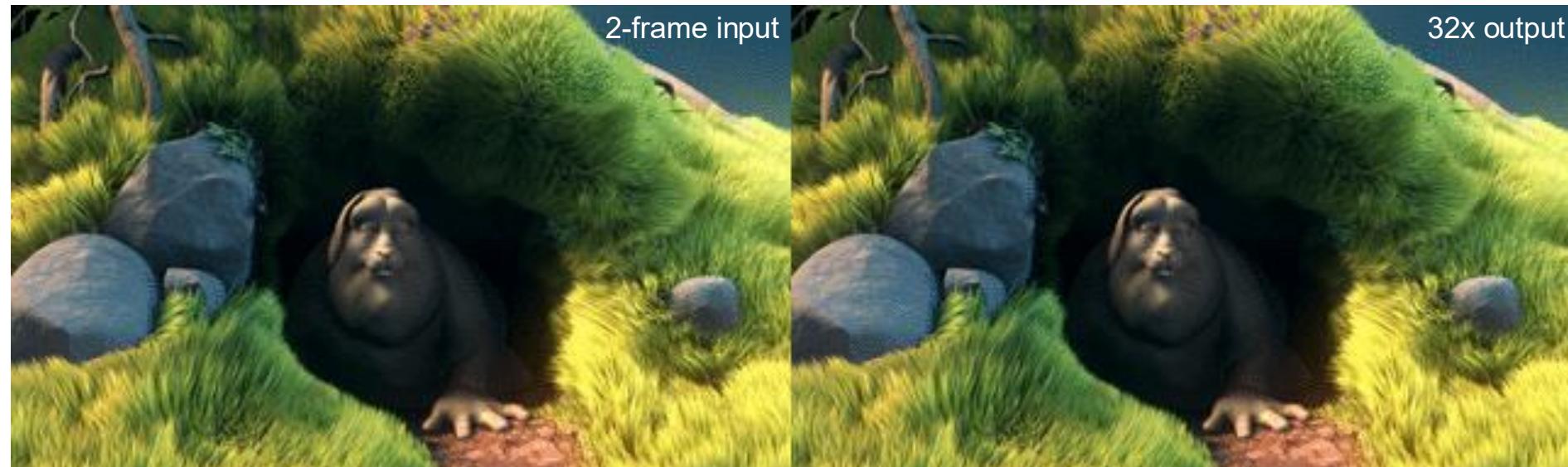
Optical Flow – Applications

(Ego-Pose) Tracking



Optical Flow – Applications

Motion Interpolation



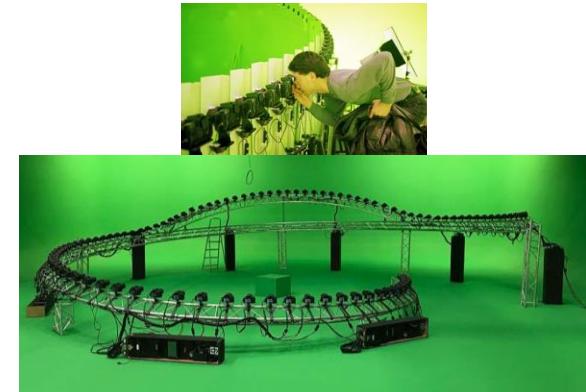
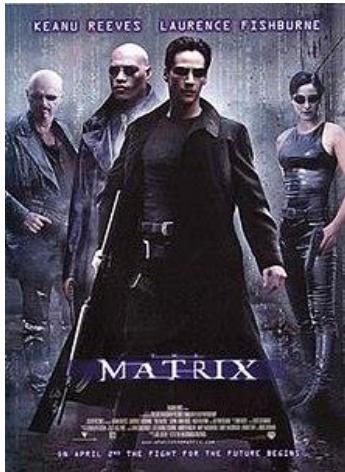
Super SloMo

Jiang, Sun, et al. CVPR 2018 Spotlight]

Incorporated into **NVIDIA NGX SDK** for the Turing GPU

Optical Flow – Applications

Pose Interpolation



The Maatrix, 1998
<https://www.imdb.com/title/tt0133093>

Academy award, 2000
Best Visual Effects



Optical Flow → Correspondence between camera views
Smooth interpolation (virtual camera) between views

Optical Flow – Applications

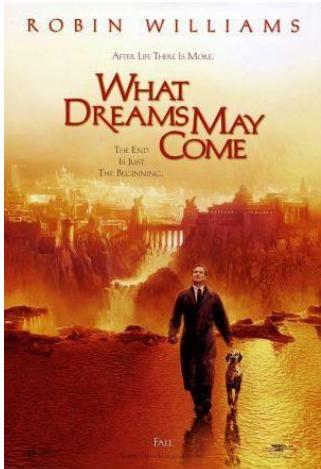
Appearance Transfer



The final shot was enabled with extensive development of [tracking techniques](#), [optical flow](#) and a specialized [particles tool](#) to produce the [painterly effects](#)

Optical Flow – Applications

Appearance Transfer



What Dreams May Come, 1998
<https://www.imdb.com/title/tt0120889>

Academy award, 1999
Best Visual Effects



Impressionist effect

Transfer: Real-world motion \leftrightarrow Painting



Same principle for
propagating annotations
across video

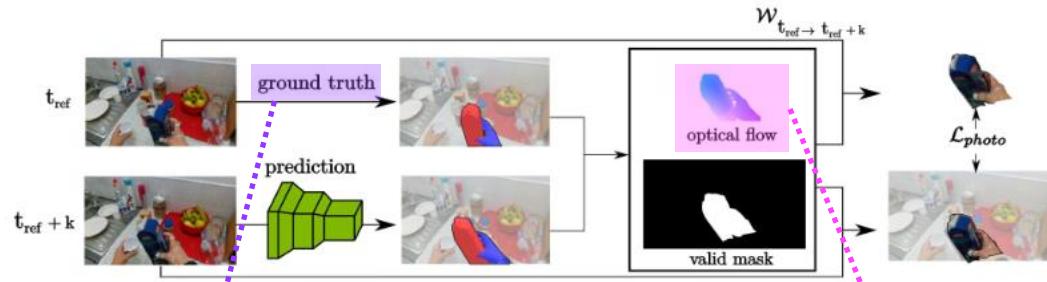
Optical Flow – Applications

Weak/Self-Supervision



LASR: Learning Articulated Shape Reconstruction from a Monocular Video [Yang et al., CVPR'21]

'Why should *motion* be treated as a *first-class citizen*? Besides that *optical flow* naturally encodes *correspondences*, it provides more *fine-grained information* than *keypoints* as well as *semantic parts*. [...] can be obtained more reliably over two consecutive frames.'



Leveraging Photometric Consistency over Time for Sparsely Supervised Hand-Object Reconstruction [Hasson et al. CVPR'20]

'[...] *annotations* are only available for a *sparse subset of frames* in a video'

'Given our estimated reconstructions, we differentiably render the *optical flow* between pairs of adjacent images and use it within the network to warp one frame to another. We then apply a *self-supervised photometric loss* that relies on the *visual consistency* between nearby images.'

'The loss effectively *propagates information* from a *few annotated frames* to the *rest of the video*.'

Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking



Tracking Demo



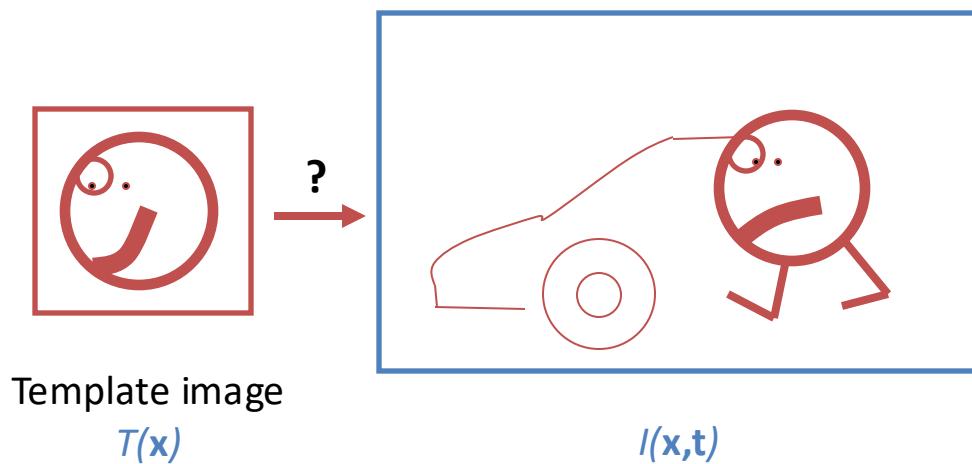
Source : Qiang Wang et. al.

Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

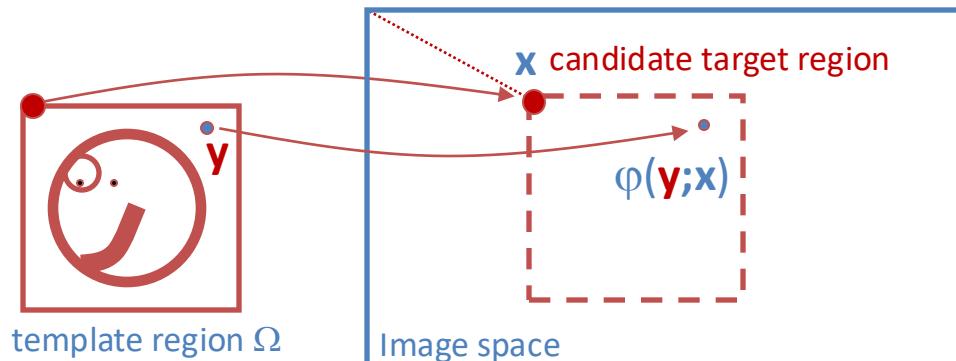
Template-based Tracking

- Given a **template** image $T(x)$ that shows a target object
- Search for the target object in an image frame $I(x,t)$...
- ... by **comparing** the **template** image $T(x)$ with image areas
- Assume that the **template** is **fixed** and **given in advance**



Template-to-Image Alignment

- ‘Align’ template into candidate target region in image via transformation: $\varphi(\mathbf{y})$
- This transformation depends on the running position \mathbf{x} of candidate region
- So, we write $\varphi(\mathbf{y}; \mathbf{x})$



Example

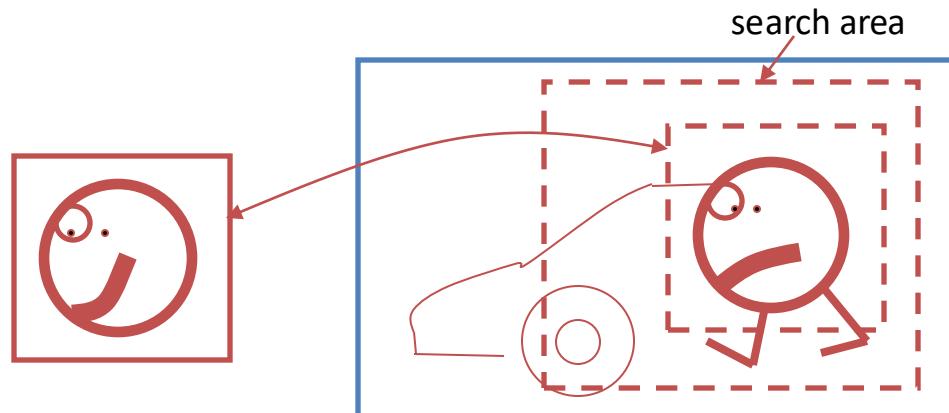
$$\mathbf{y} = [2, 10]$$

$$\mathbf{x} = [4, 06]$$

$$\varphi(\mathbf{y}; \mathbf{x}) = [6, 16]$$

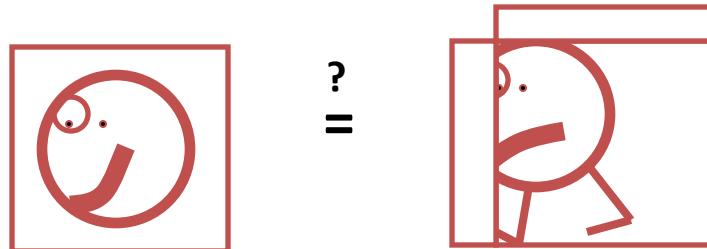
Search Space

- ‘Align’ the template with every possible candidate region (i.e. every possible position y) in the image
- Find the **most similar candidate** according to a **similarity measure**
- **Search** only in area around previous position (assume: object won’t have moved far)



Similarity Measure

- Need a measure of **how similar** the template and **cand date** are
- The similarity measure can be based on:
 - Pixelwise **intensity/color difference** → **SSD** and **Correlation** trackers
 - **Histogram difference** → **Mean-shift** tracker



SSD and Correlation

- SSD is short for ‘**Sum of Squared Differences**’:

$$SSD(\mathbf{y}) = \sum_{\mathbf{x} \in \Omega} \|I(\mathbf{x} + \mathbf{y}) - T(\mathbf{x})\|^2 \longrightarrow y^* = \min_{\mathbf{y}} SSD(\mathbf{y})$$

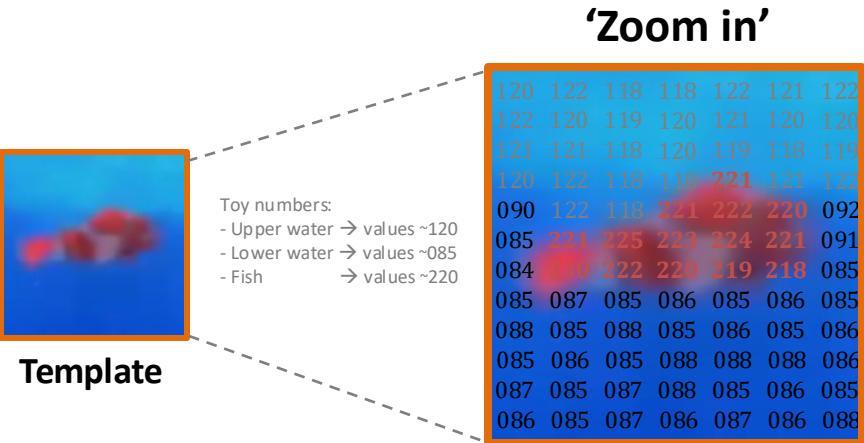
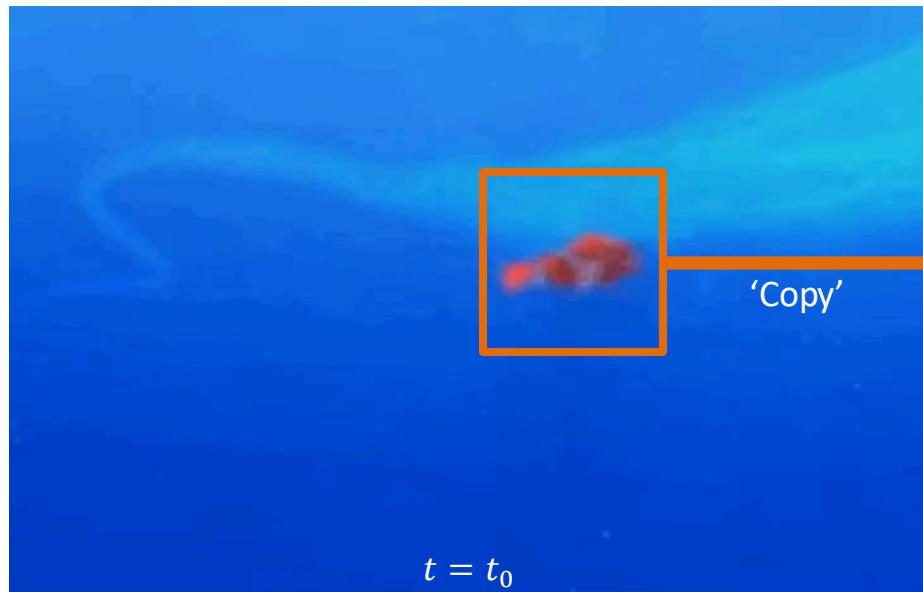
- Simpler measure → **Cross-Correlation** (unnormalized):

$$CC(\mathbf{y}) = \sum_{\mathbf{x} \in \Omega} I(\mathbf{x} + \mathbf{y})T(\mathbf{x}) \longrightarrow y^* = \max_{\mathbf{y}} CC(\mathbf{y})$$

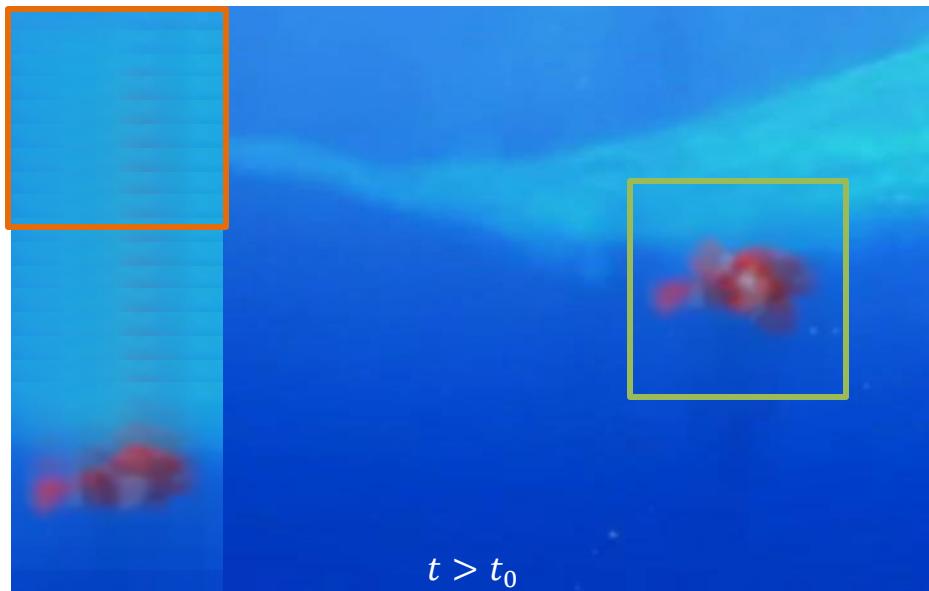
Exhaustive Search

- Calculate **SSD** or **CC** for every candidate position **y** in a search window
- Choose the position with the **smallest SSD** or the **biggest CC**
- **Strengths:**
 - Robustness
 - Simplicity in implementation
- **Weaknesses:**
 - Computations **time-consuming** for **large search window**
 - Only suitable for **translation**

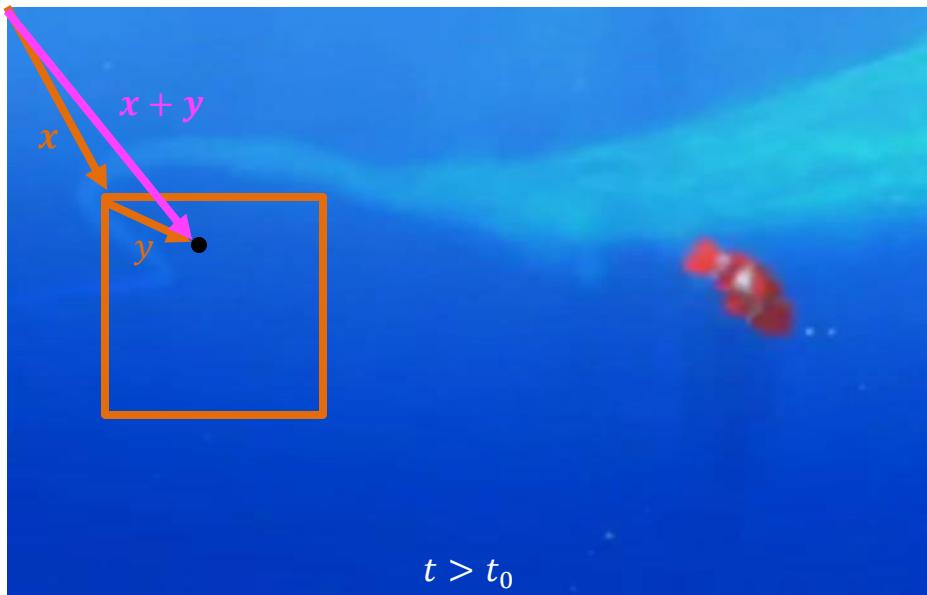
Toy Example – Define Template



Toy Example – ‘Swipe’ Template over Image



Toy Example – Similarity Metric



$$SSD(x, t) = \frac{1}{M} \sum_{y \in \Omega} (I(\mathbf{x} + \mathbf{y}, t) - T(\mathbf{y}))^2$$

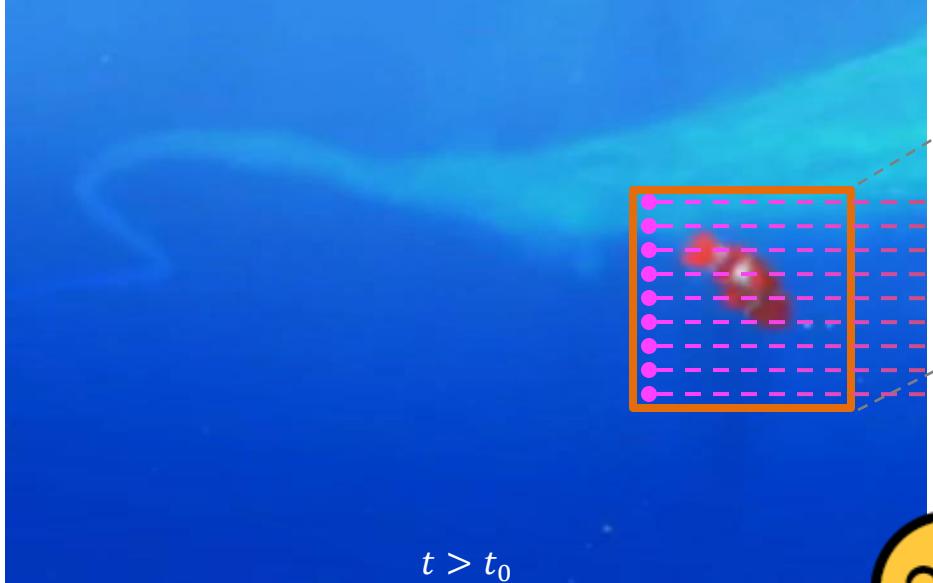
Toy Example – Similarity Metric – Pixel-wise



$$SD(x, t) = \frac{1}{M} \sum_{y \in \Omega} (I(x + y, t) - T(y))^2$$

Toy Example – Similarity Metric – Pixel-wise

CV



$t > t_0$

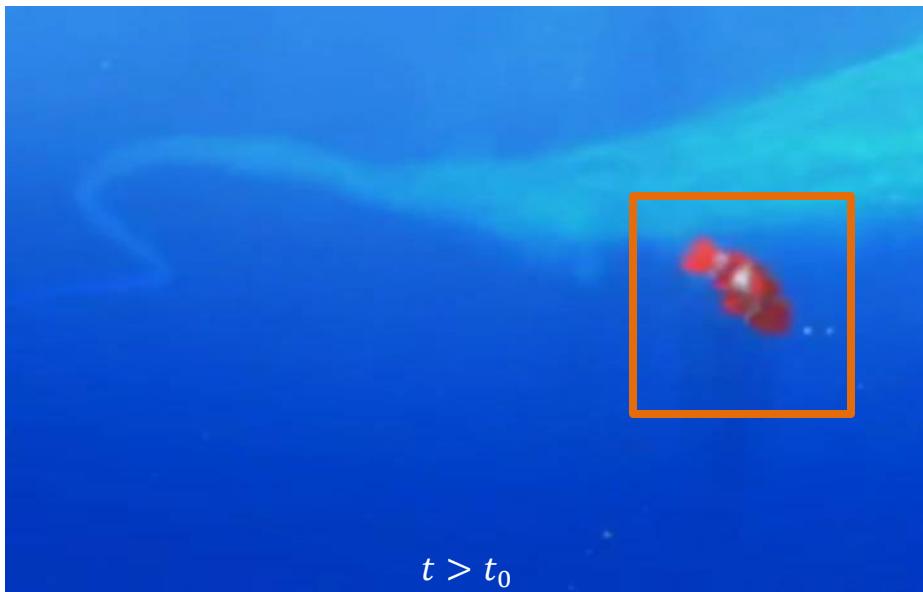


$$SD(x, t) = \frac{1}{M} \sum_{y \in \Omega} (I(x + y, t) - T(y))^2$$

23	124	120	120	121	123	124
24	122	116	117	118	117	116
21	120	224	119	116	117	116
00	119	120	117	122	121	123
090	092	224	221	222	095	092
085	087	086	223	224	098	091
084	086	087	220	219	086	085
085	087	085	086	205	086	085
088	085	088	085	086	085	086
085	086	087	086	085	086	084
084	085	084	085	085	083	085
084	083	087	086	086	086	087

120	122	118	118	122	121	122
122	120	119	120	121	120	123
121	121	118	120	119	118	119
120	122	116	119	221	121	122
090	122	118	221	222	220	092
085	224	225	223	224	221	091
084	090	222	220	219	218	085
085	087	085	086	085	086	085
088	085	088	085	086	085	086
085	086	085	088	088	088	086
087	085	087	088	085	086	085
086	085	087	086	087	086	086

Toy Example – SSD



Small SSD == Small difference

$$x^*(t) = \arg \min_x SSD(x, t)$$

★ optimal value



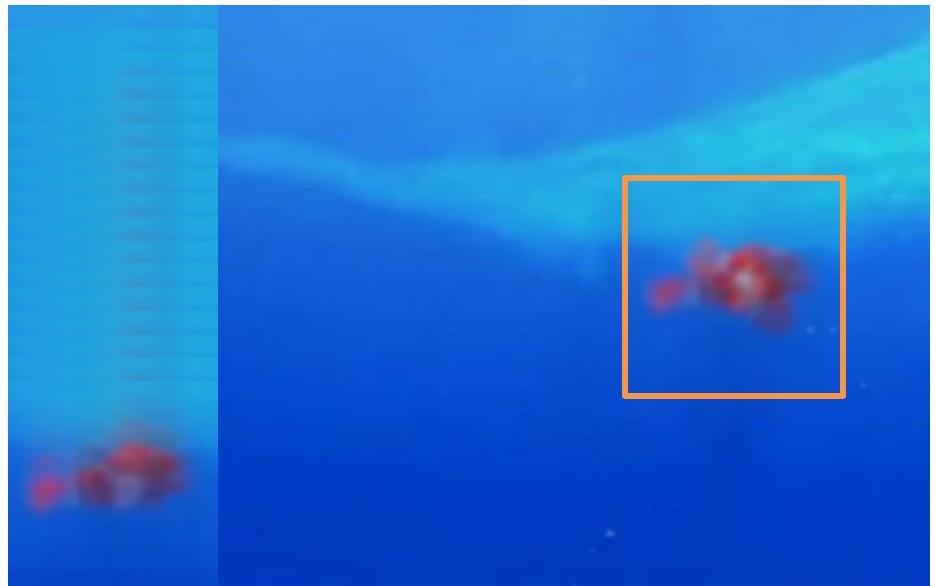
Or maximize the
Correlation metric

Toy Example – Street Smart

‘Global’ search works *well*
but is computationally *expensive*

If we know the motion history of the fish
(i.e. we have tracked the fish for a few frames) ...?

- *Predict* future motion &
- *Restrict* search in ‘*local*’ area



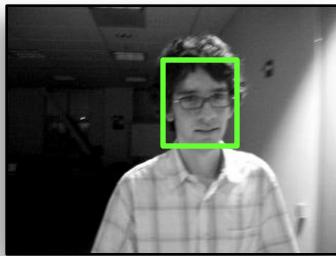
$$t > t_0$$

Outline

- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Tracking by Detection

First frame is labeled



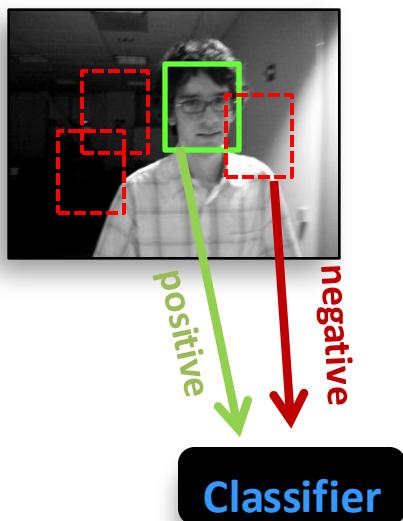
Classifier



Online classifier (i.e. Online AdaBoost)

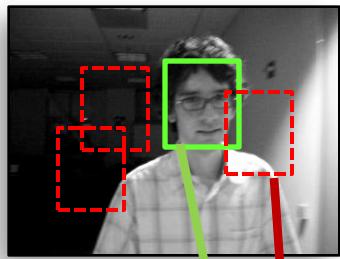
Tracking by Detection

Train/Update model w. 1 **positive** & some **negative** patch



Tracking by Detection

Get next frame



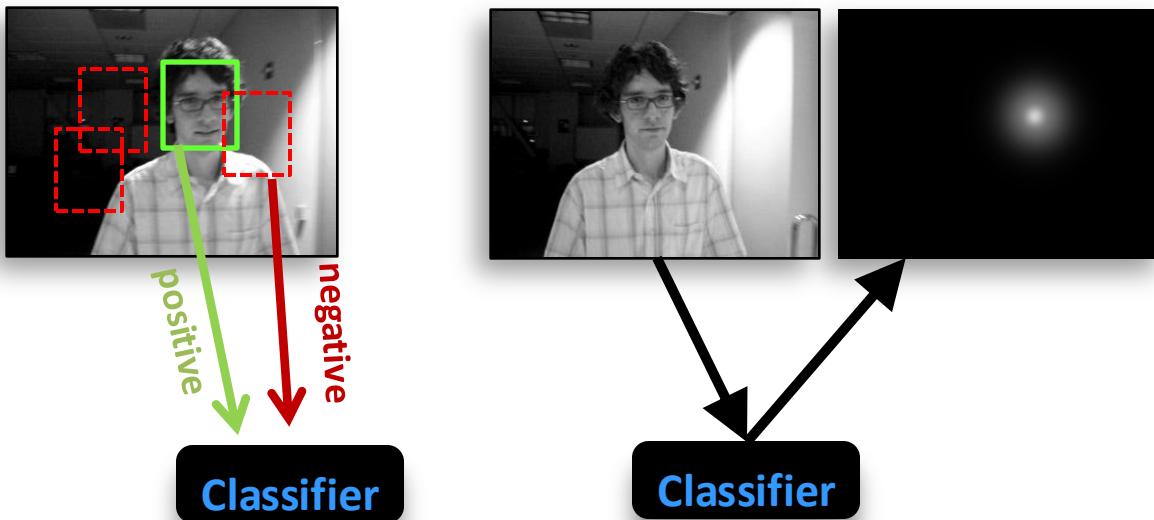
positive
negative

A diagram illustrating the feedback loop of tracking by detection. Two arrows point downwards from the frame above. A green arrow on the left is labeled "positive", pointing to a black button labeled "Classifier". A red arrow on the right is labeled "negative", pointing to the same black button. This indicates that the classifier provides both positive (correct) and negative (incorrect) feedback to refine its detections in the next frame.

Classifier

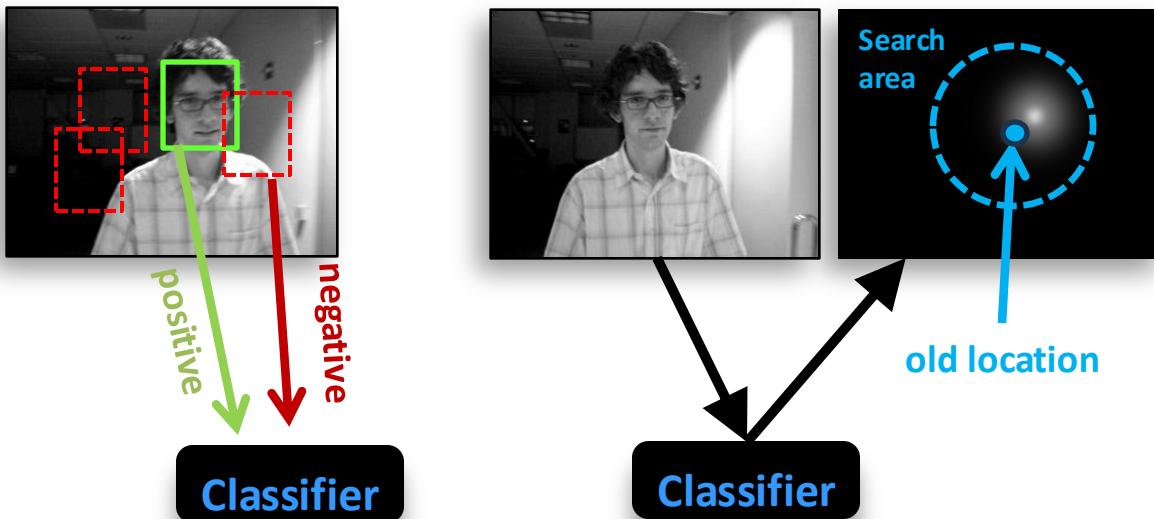
Tracking by Detection

Evaluate classifier in some **search window**



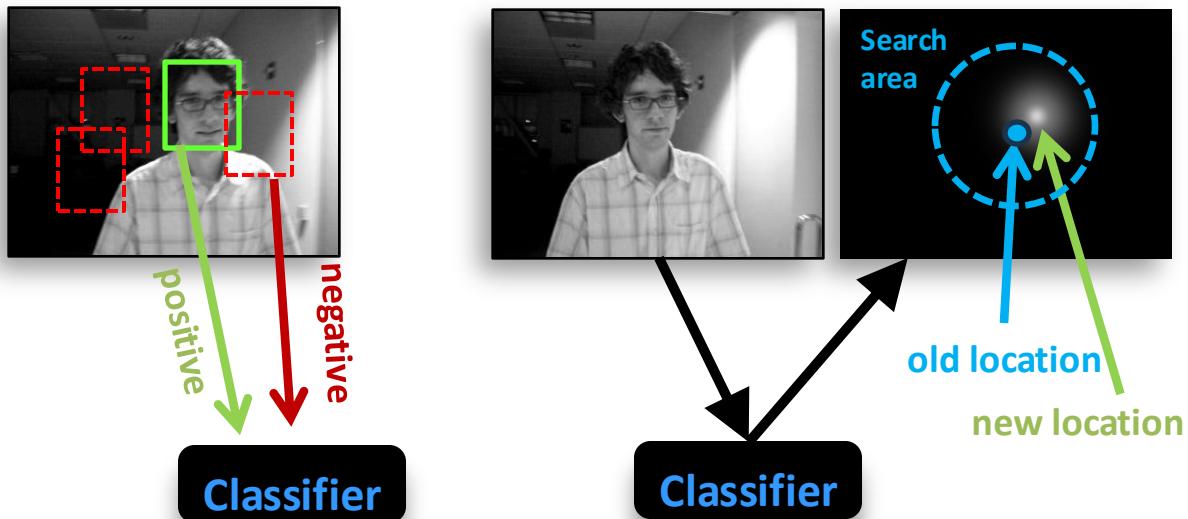
Tracking by Detection

Evaluate classifier in some **search window**



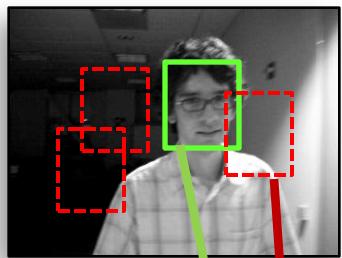
Tracking by Detection

Find max response



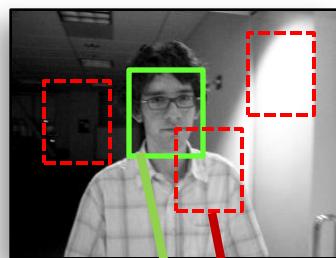
Tracking by Detection

... Repeat ...



positive
negative

Classifier



positive
negative

Classifier

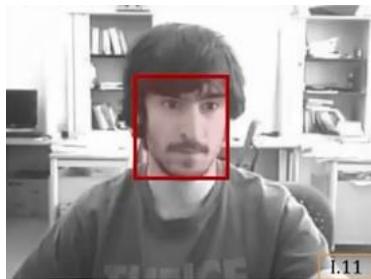
...

Tracking by Detection

Most tracking works:

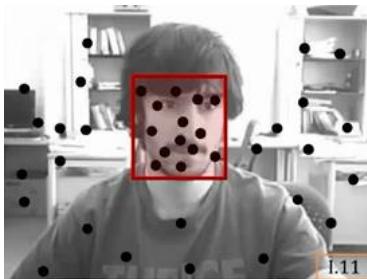
- Focus on **appearance** model
- Borrow techniques from **object detection** (see: next week)
 - ‘Slide’ a **discriminative classifier** around image
- **Adaptive** appearance model

Tracking by Detection – SIFT



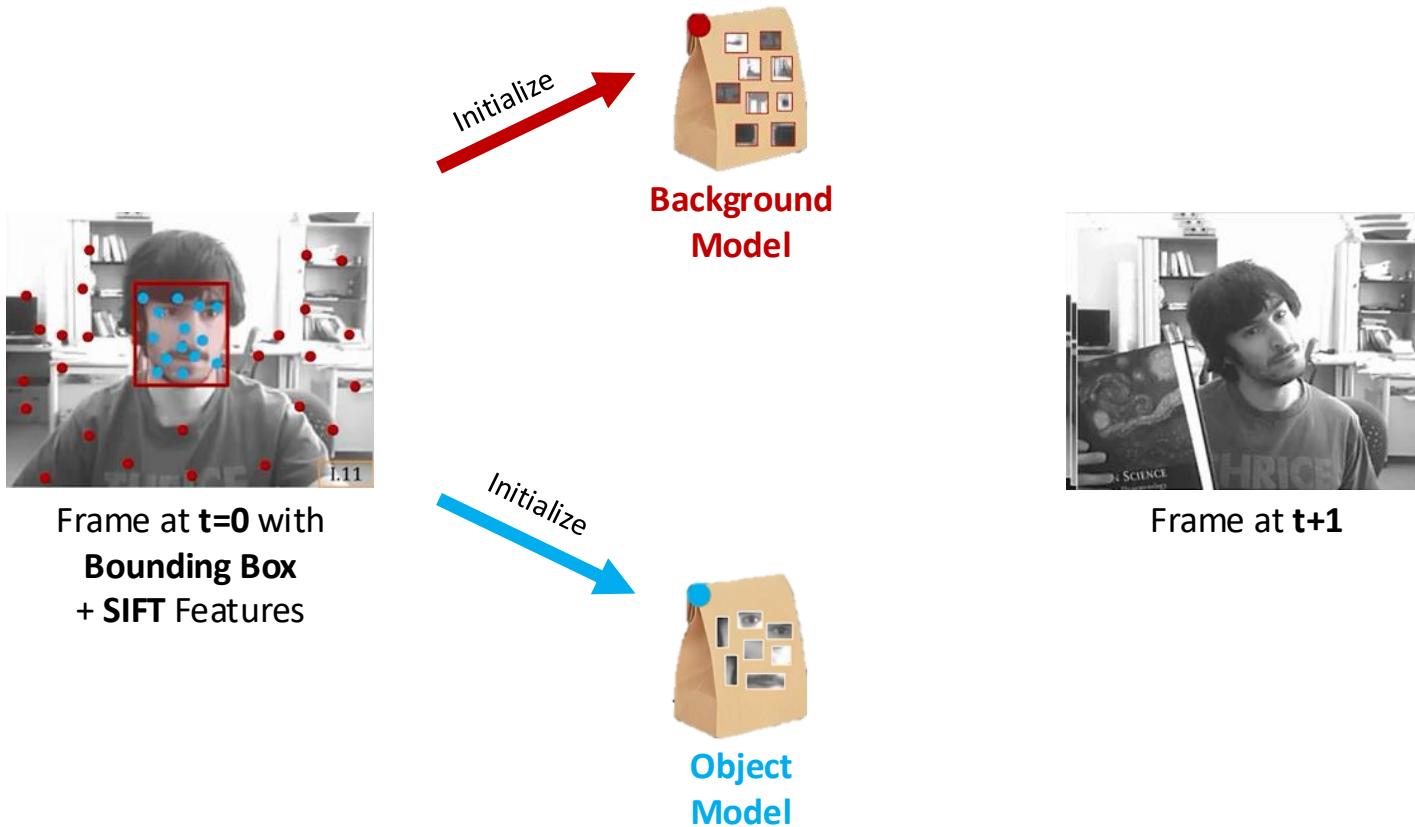
Frame at **t=0** with
Bounding Box

Tracking by Detection – SIFT

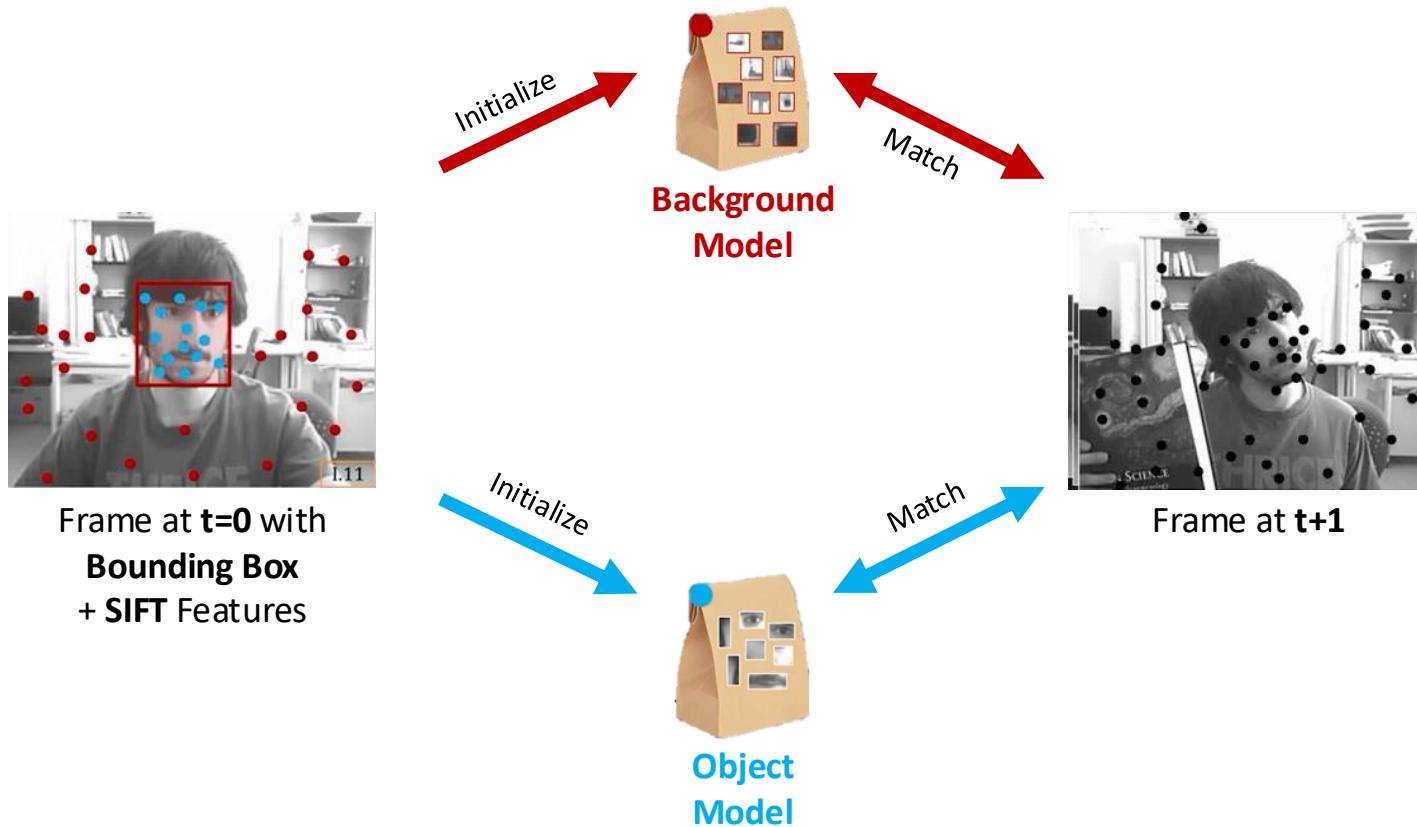


Frame at **t=0** with
Bounding Box
+ **SIFT** Features

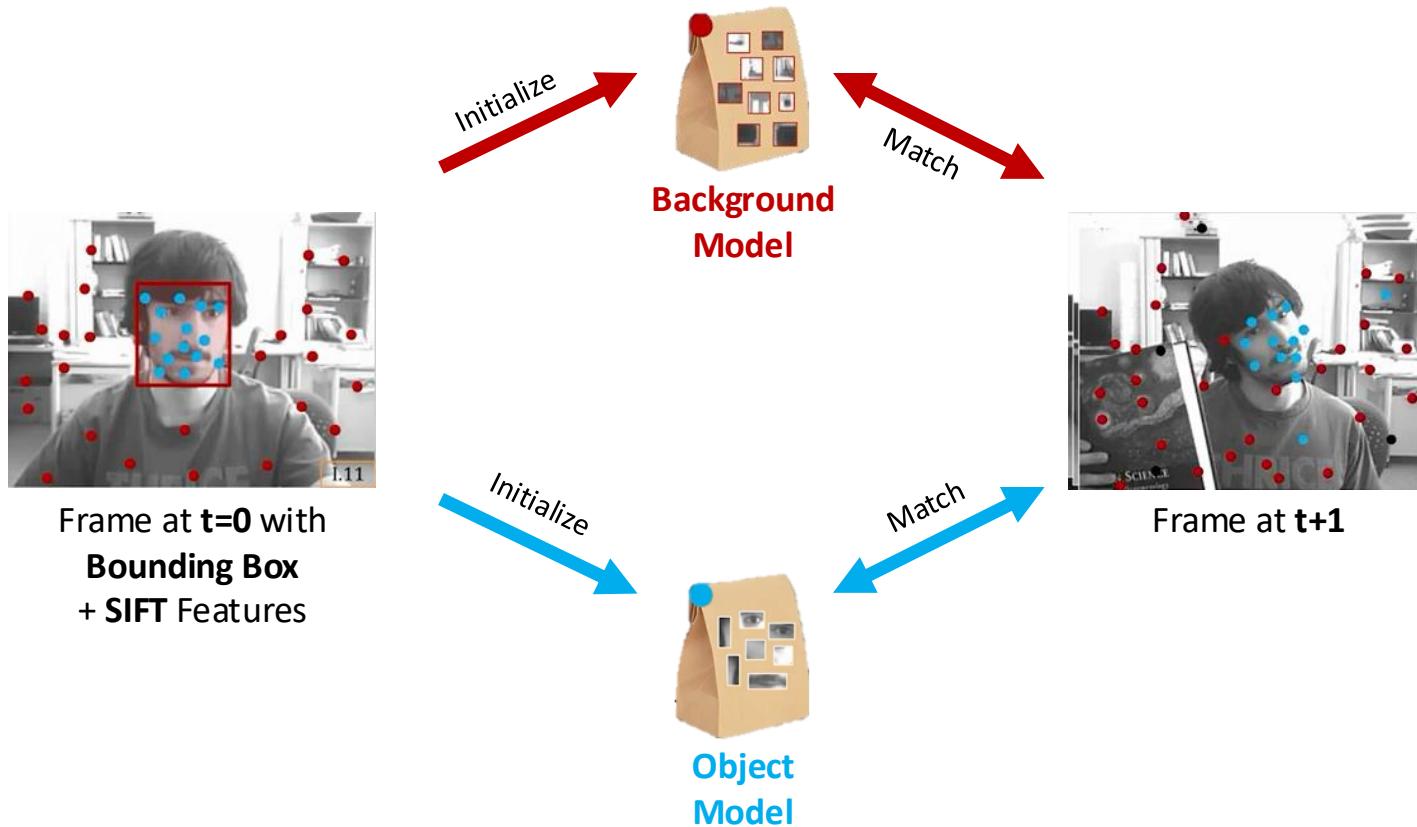
Tracking by Detection – SIFT



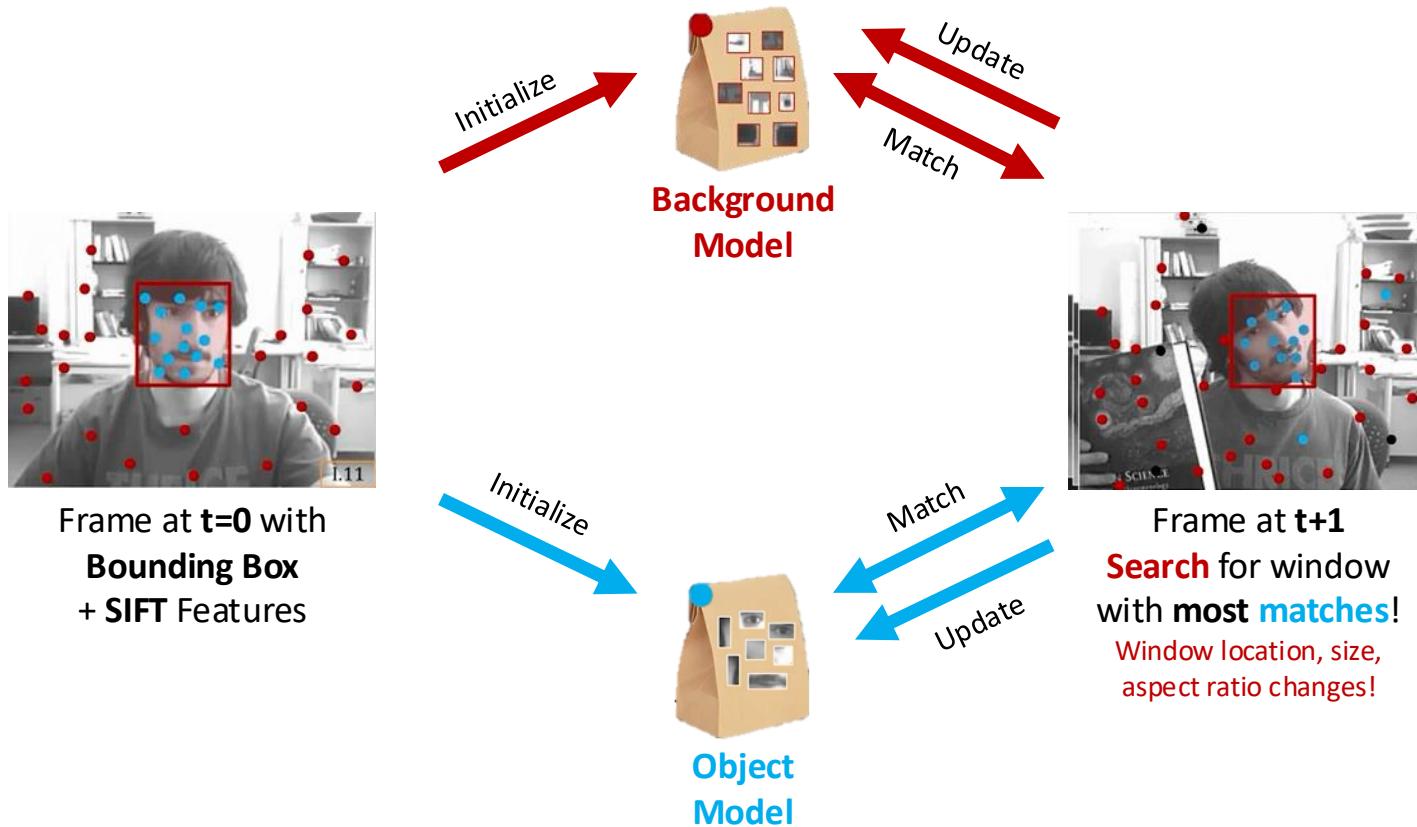
Tracking by Detection – SIFT



Tracking by Detection – SIFT



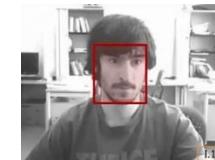
Tracking by Detection – SIFT



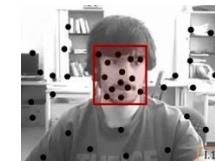
Tracking by Detection – SIFT

Frame at $t=0$

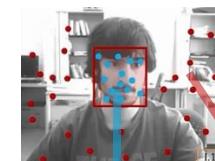
- User selects a **Bounding Box W_1** as object/target



- Compute SIFT features (or similar) for the frame



- Classify features in Box as Object & assign to set O_0



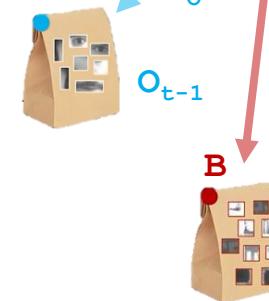
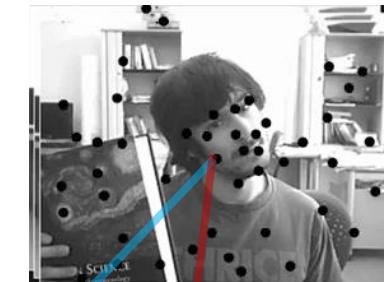
- Classify other features as BackGround & assign to B



Tracking by Detection – SIFT

Frame at $t > 0$

- Compute SIFT features & descriptors $\{v_1, \dots, v_k\}$
- For each feature & corresponding descriptor v_i
 - Compute distance d_o between v_i and the best match in object set O_{t-1}
 - Compute distance d_B between v_i and the best match in object set B
 - $C(v_i) = \begin{cases} +1 & \text{if } d_o/d_B < 0.5 \text{ (v_i belongs to object)} \\ -1 & \text{otherwise (v_i does not belong to object)} \end{cases}$

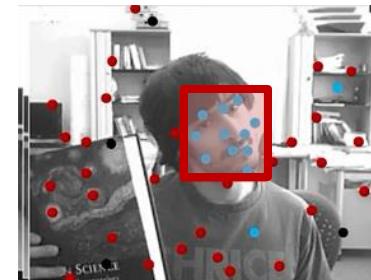
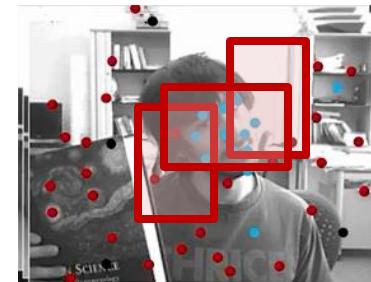


Continues at
next slide

Tracking by Detection – SIFT

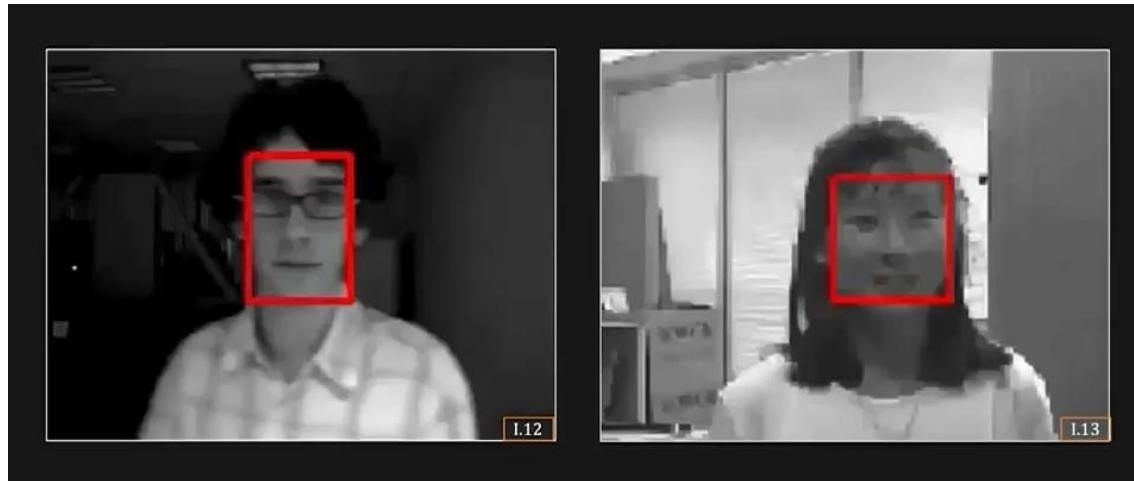
Continued from
previous slide

- For each **Search Box W**
 - Compute $\varphi(W) = \sum C(v_i)$ for all features v_i in W
 - Compute a heuristic $\tau(W, W_{t-1})$ that **penalizes large deviations** from location, size, shape of W_{t-1}
 - Compute **Match Score:** $\mu(W) = \varphi(W) - \tau(W, W_{t-1})$
- Select Box W_t with **best Match Score** as new object location
- Update object appearance model:
 $O_t = O_{t-1} \cup \{v_i\}$ with all v_i in W_t with $C(v_i) = +1$



Tracking by Detection – SIFT

More robust to
occlusion, scale &
orientation changes
than template-based
tracking

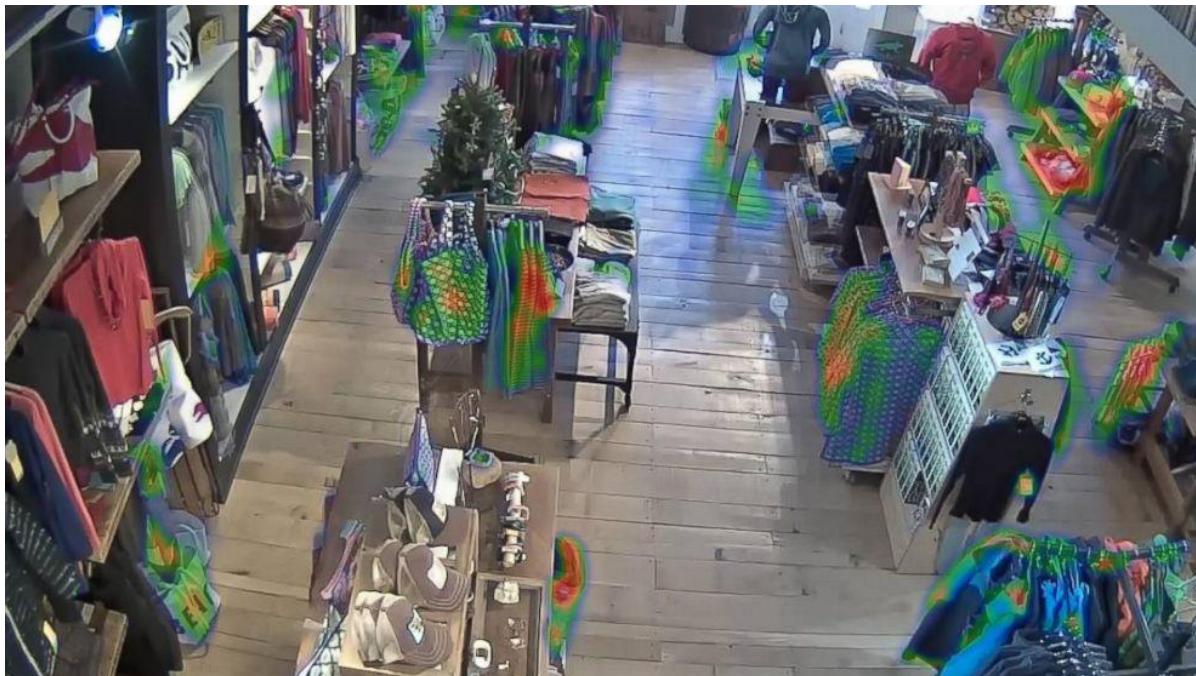


Tracking by Detection – SIFT



Tracking People in the Wild

Tracking by Detection – SIFT



Customer Behavior – In-store Analytics

Outline

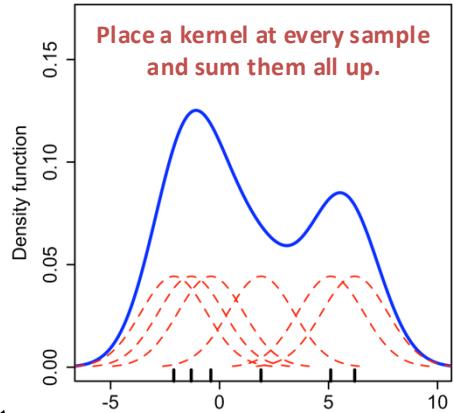
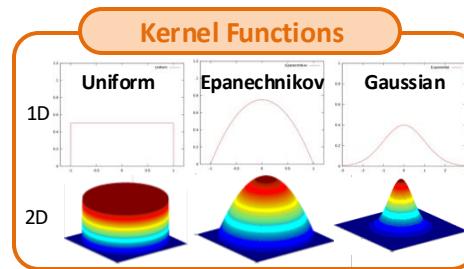
- Motion
- Optical Flow (OF)
 - Aperture Problem
 - Lukas-Kanade OF
 - Coarse-to-fine OF Estimation
 - Applications
- Tracking
 - Template Tracking
 - Tracking by Detection
 - Mean-shift Tracking

Detour: Kernel Density Estimation (KDE)

Idea: Estimate a probability distribution of a given point set.

Compose the distribution as a sum of n basis/kernel functions K .

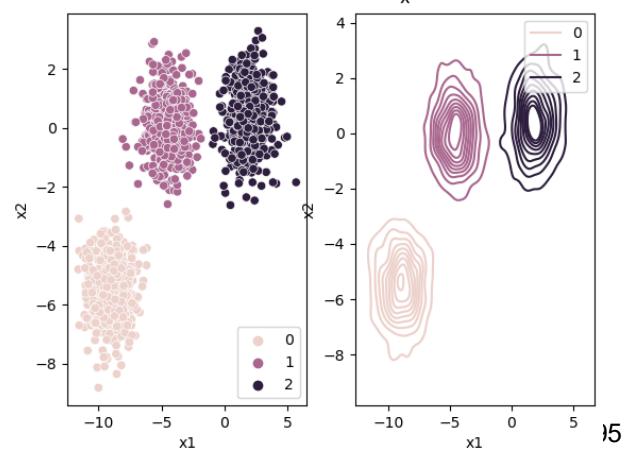
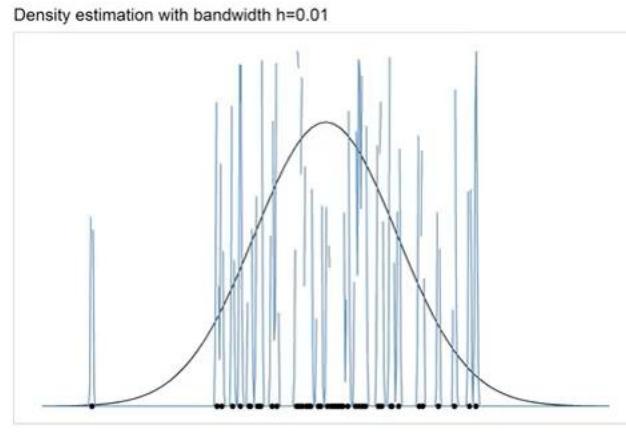
$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$



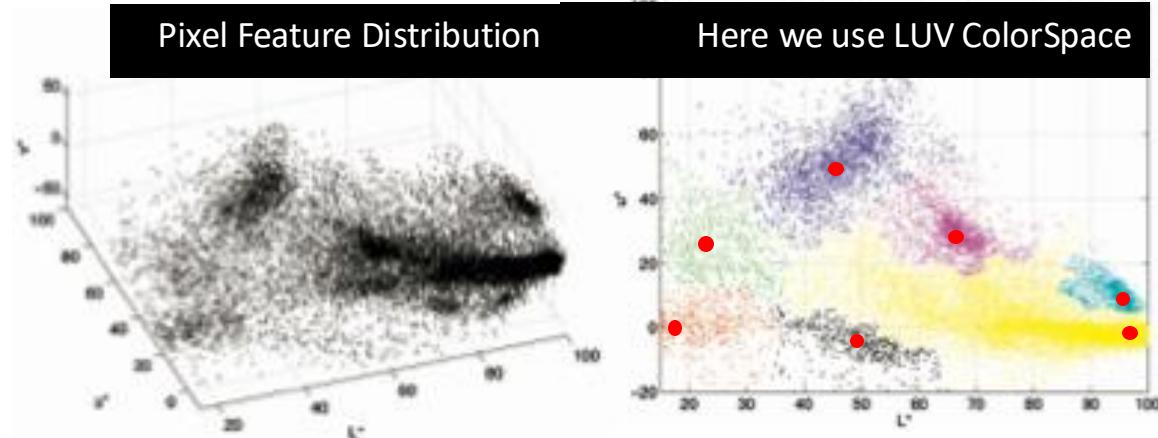
Bandwidth h :

Width of kernel
(sigma for Gaussian)

Defines level of detail
and number of modes.

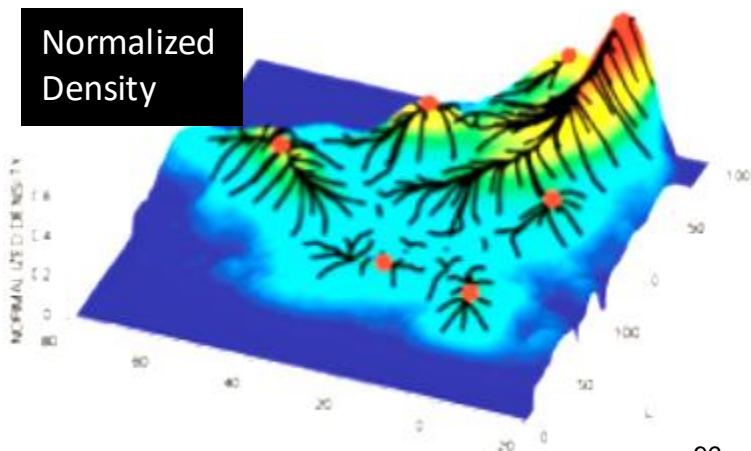


Mean Shift – Segmentation / Clustering

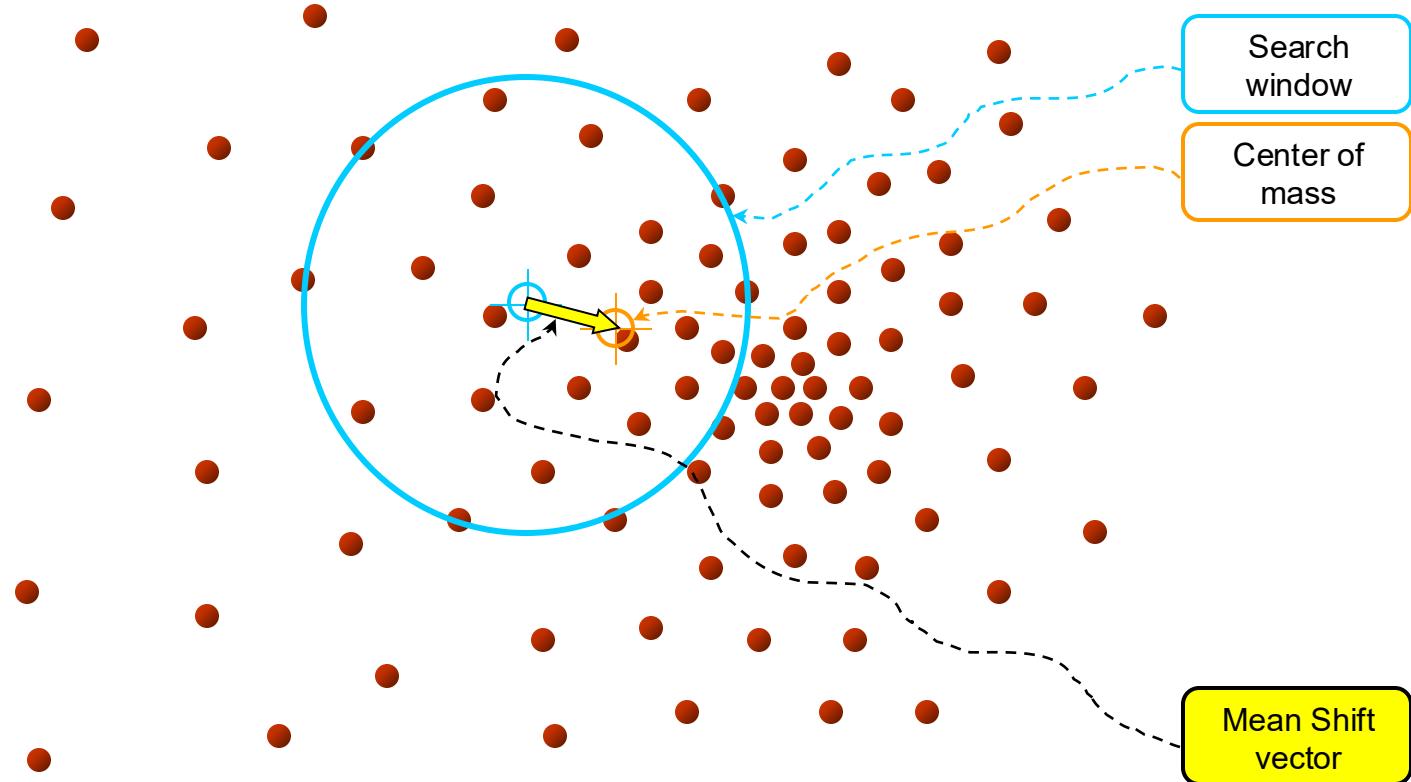


Idea: Density-based clustering

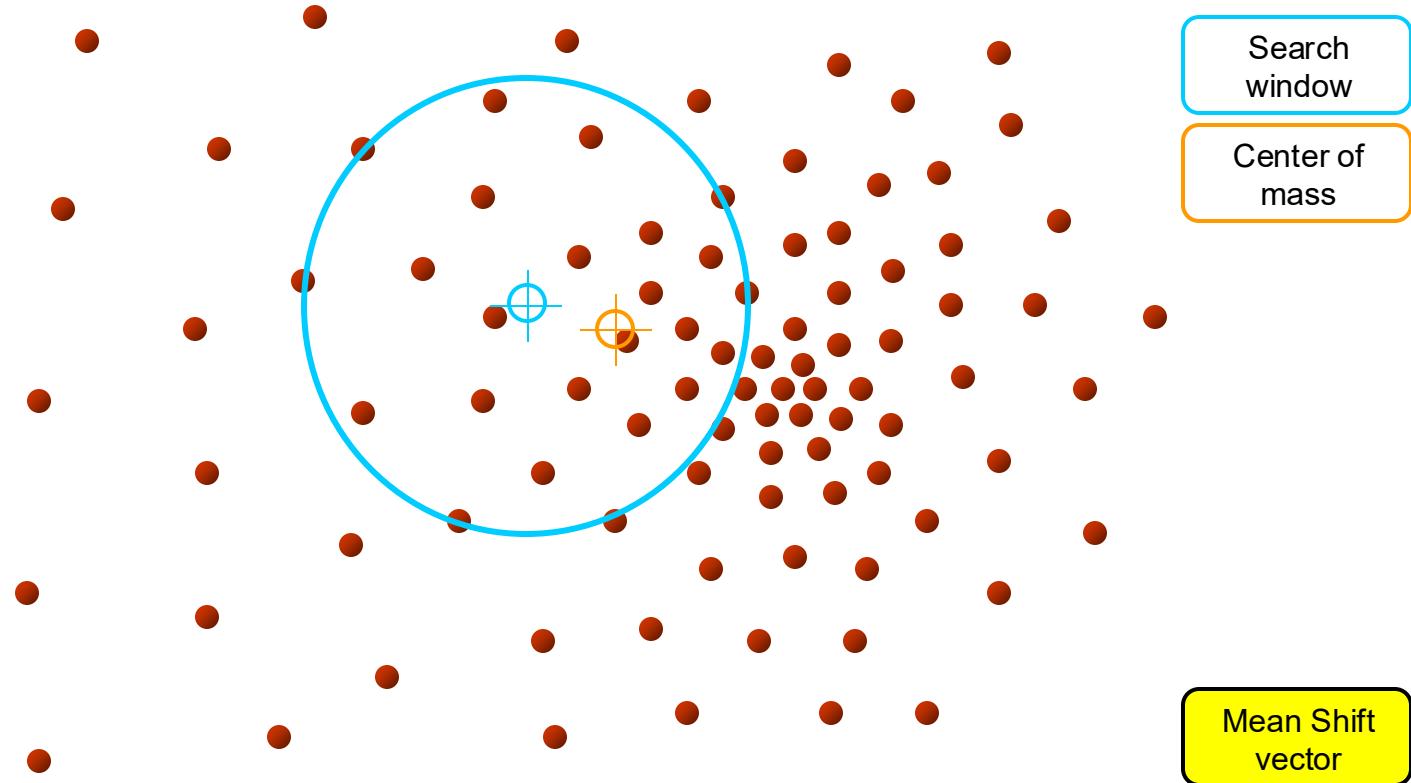
Goal: Find all modes of a given probability distribution



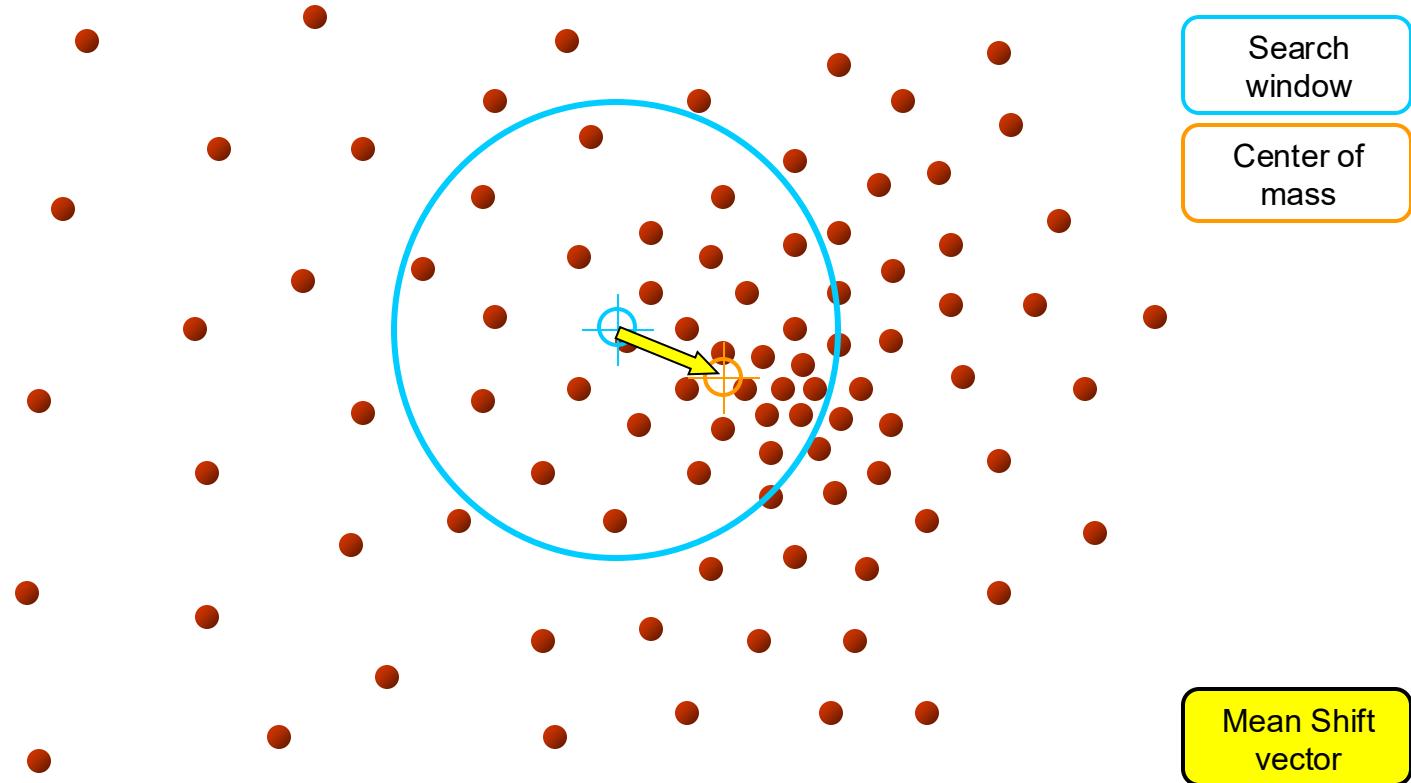
Mean Shift Algorithm – 2D Toy Example



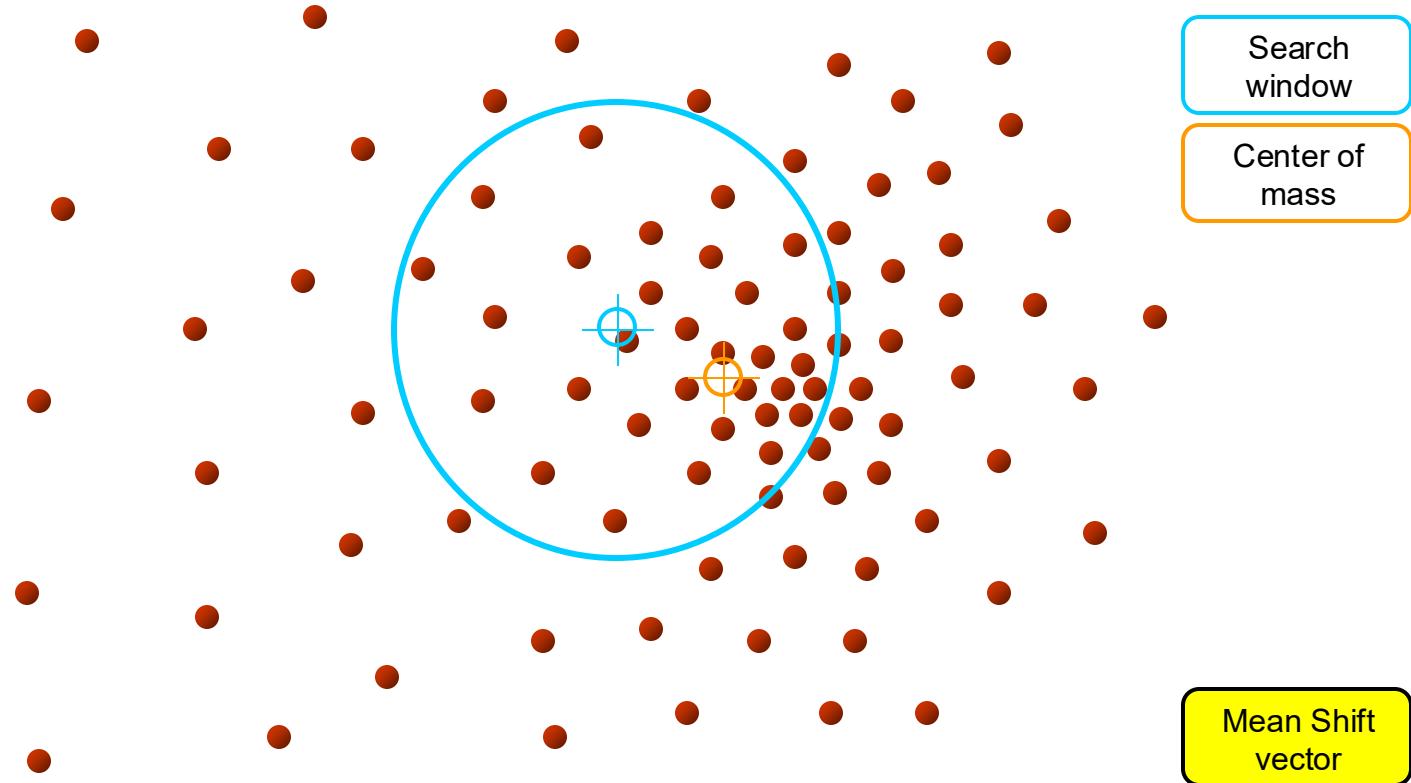
Mean Shift Algorithm – 2D Toy Example



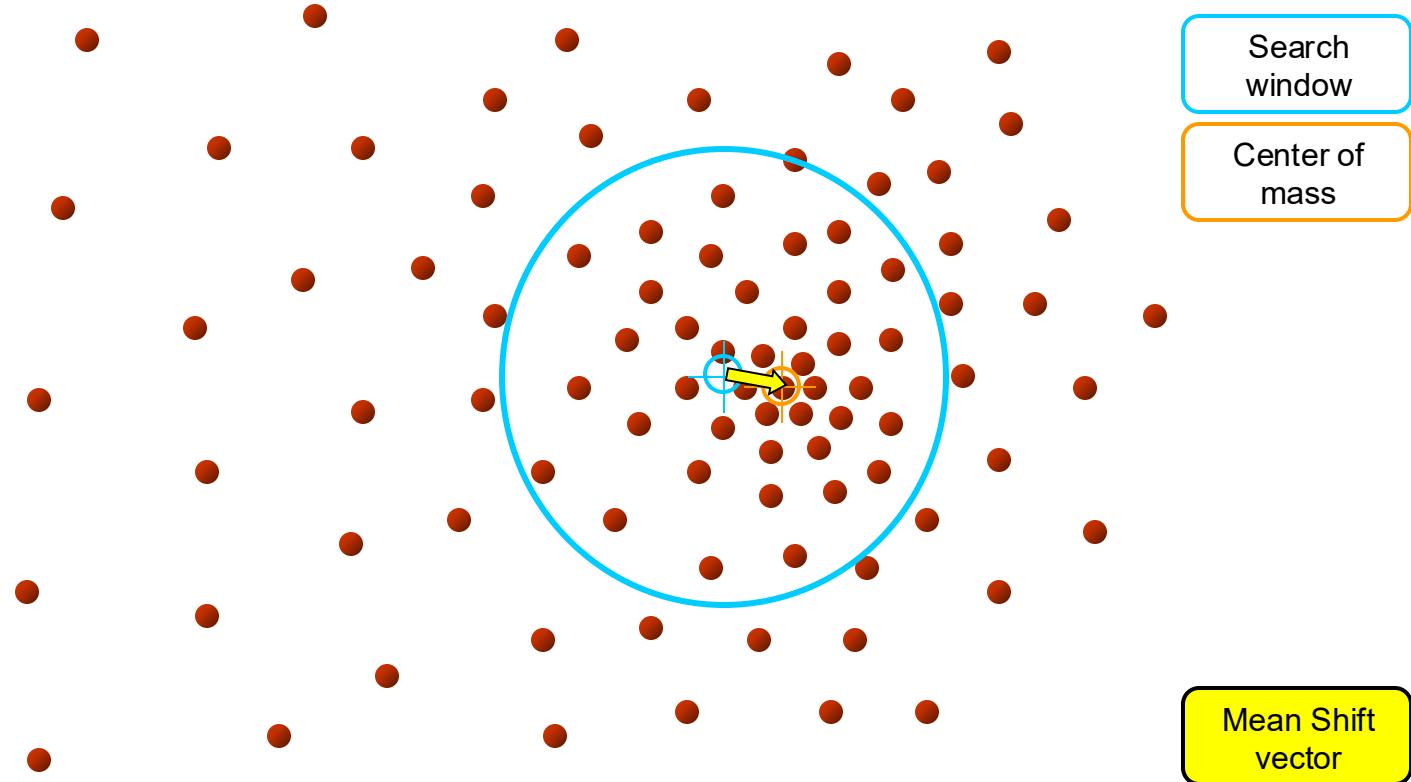
Mean Shift Algorithm – 2D Toy Example



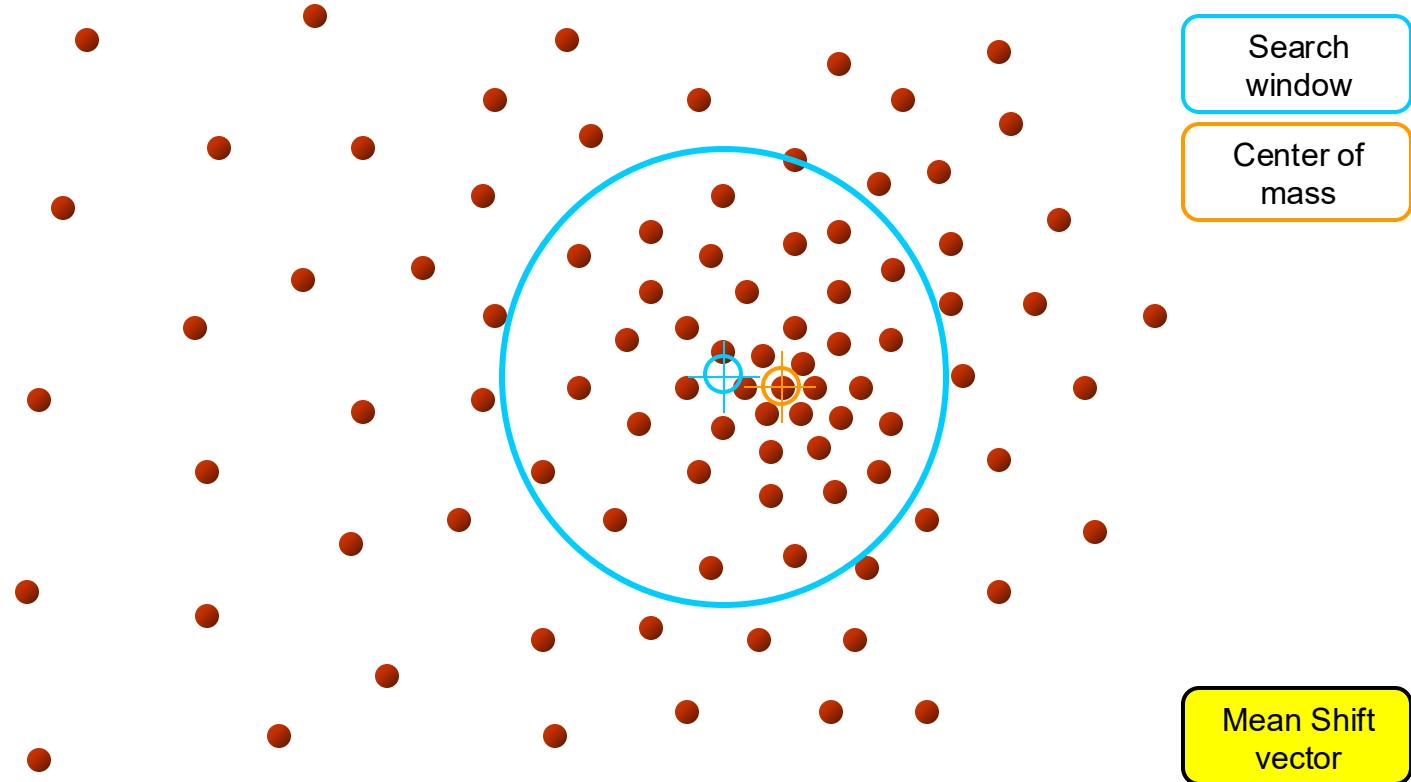
Mean Shift Algorithm – 2D Toy Example



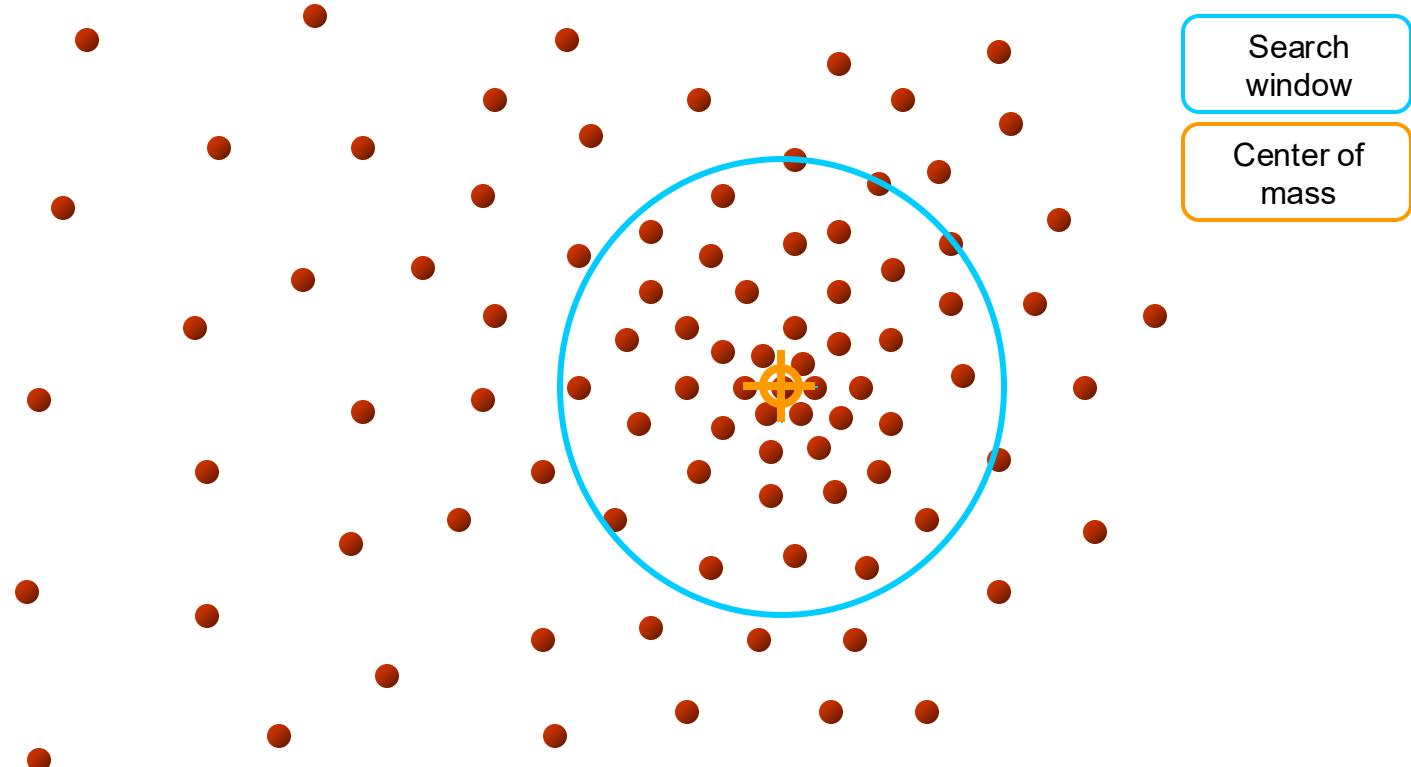
Mean Shift Algorithm – 2D Toy Example



Mean Shift Algorithm – 2D Toy Example



Mean Shift Algorithm – 2D Toy Example

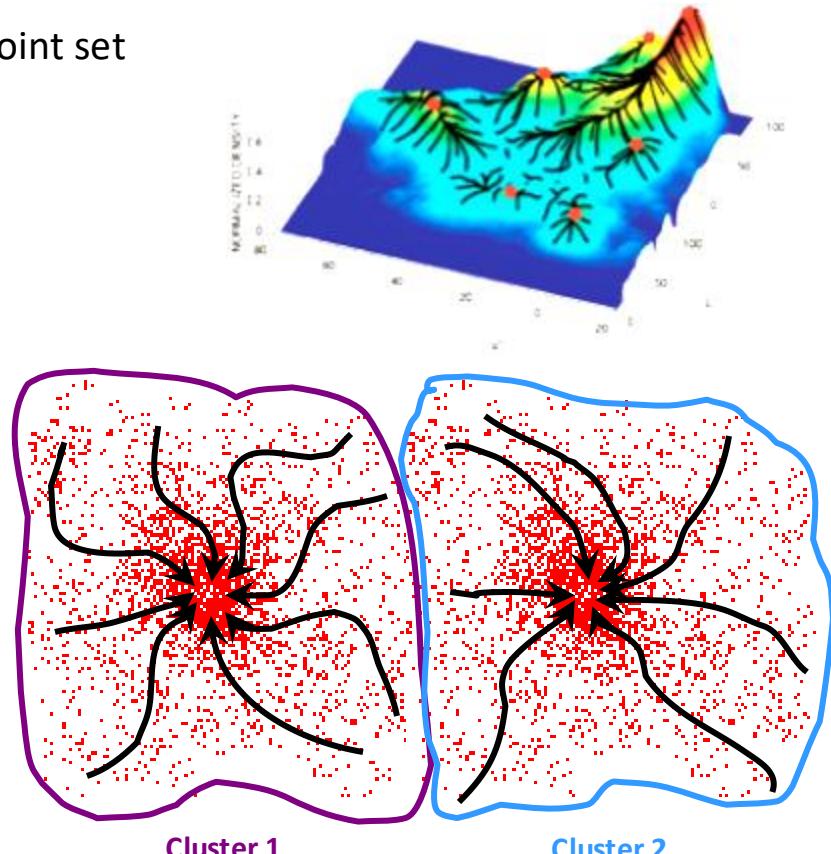


Mean Shift Algorithm

The mean shift algorithm seeks the **modes** of the given point set

1. Choose kernel and bandwidth
2. For each point:
 - a) Center a window with a kernel on that point
 - b) Compute the mean of the data in the search window
 - c) Center the search window at the new mean location
 - d) Repeat (b,c) until convergence
3. Assign points that lead to nearby modes to the same cluster

- **Attraction basin:** Region for which all trajectories lead to the same mode
- **Cluster:** All points in the attraction basin of a mode
- **Note:** Bandwidth and kernel type controls the size of attraction basins and the number of modes; and thus eventually the number of found clusters.



[Figure from: Y. Ukrainianitz & B. Sarel]

Mean Shift Segmentation

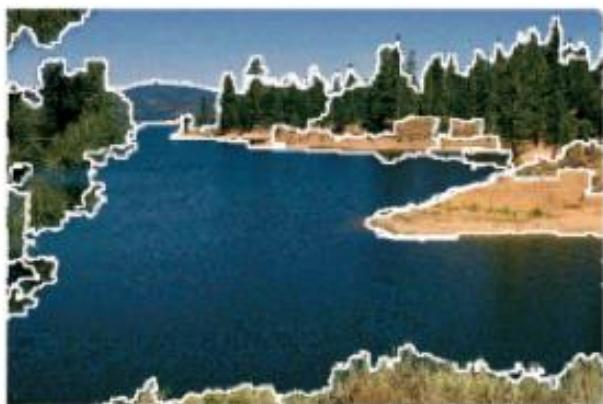
Results of clustering
in LUV color space.



[Comaniciu, Meer. [Mean shift: A robust approach toward feature space analysis](#), TPAMI, 2002]

Mean Shift Segmentation

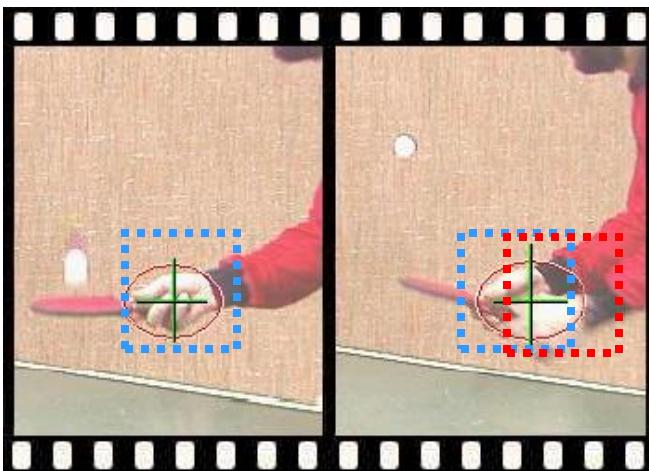
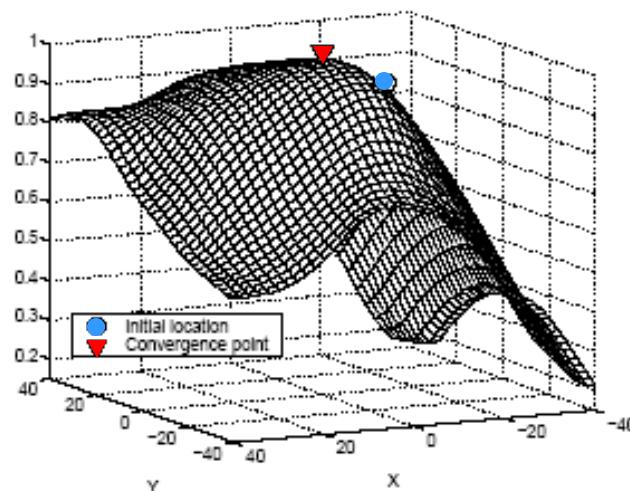
Results of clustering
in LUV color space.



[Comaniciu, Meer. [Mean shift: A robust approach toward feature space analysis](#), TPAMI, 2002]

Mean Shift Tracking

- Define a **target region**: $\vec{q} = (q_1, \dots, q_m)$ with **color distribution** from m pixels
- Next frame: Search vicinity with **candidate region**: $\vec{p}(y) = (p_1(y), \dots, p_m(y))$ at position y
- With a **similarity function** $f[\vec{p}(y), \vec{q}]$ for **comparing histograms** (e.g. cosine similarity), **mean shift** finds the **next best position** of the target candidate region

target region: \vec{q} candidate region: $\vec{p}(y)$ similarity function $f[\vec{p}(y), \vec{q}]$

In every time step:

- Use mean shift update
- to find **best location y**
- by locally maximizing the **region similarity function** starting from the previous frame position.

Mean Shift Tracking Results



Closing Remarks

Action Points

- This week:
 - Finish [Lab-2](#) assignment
 - Start [Lab-3](#) assignment – Release tomorrow

Disclaimer: might include next-Tuesday's material

Disclaimer

Many of the slides used here are obtained from online resources (including many open lecture materials) without appropriate acknowledgement. They are used here for the sole purpose of classroom teaching. All the credit and all the copyrights belong to the original authors. You should not copy it, redistribute it, put it online, or use it for any other purposes than for this course.