UniGe | MaLGa

motion analysis

Computational Vision

Francesca Odone - francesca.odone@unige.it

introduction

What do we gain if we analyse videos instead than images?

we focus on *motion information:*

- *We will now* consider low level algorithms (temporal features).
- Later in the course we will introduce higher level (action/activity classification) methods

introduction

We are observing a scene with **one** camera acquiring a set of images “close in time”

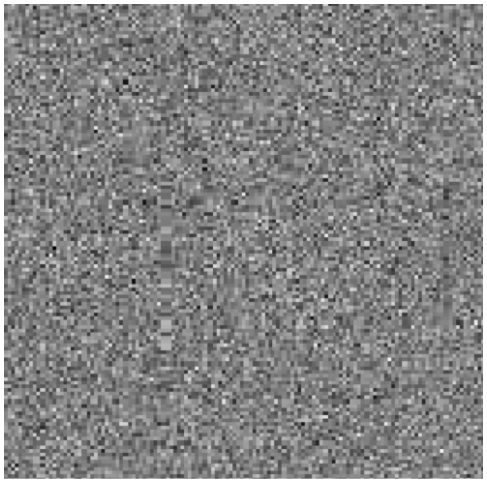
image sequence: series of N images, or *frames,* acquired at discrete time instants

$$t_k = t_0 + k\Delta t$$

fixed time interval (small)

the importance of motion cues

example



the presence of a structure is suggested to the visual system only by motion information: the changes we perceive from one frame to the next

Introduction

Brightness Constancy assumption

An important **assumption** to make when analysing image sequences is the following:

illumination does not vary in the observation interval

In this case the image changes from t_1 to t_2 are caused by the *relative motion* between scene and camera

the problems of motion

We may identify different types of questions related with motion analysis

- **correspondence:** which elements of a frame correspond to which elements of the next frame?

  (very closely related with image matching…)

  → motion estimation - optic flow computation

- **reconstruction:** what was the 3D position and 3D velocity of the observed elements?

Since adjacent frames are similar, correspondence is rather easy, but reconstruction is hard!!

the problems of motion (cont'd)

Other problems related to motion analysis

- **motion segmentation:** what regions of the image plane correspond to different moving parts? in the case the camera is still, motion segmentation is called **change detection**

- **tracking:** estimate the (2D or even 3D) trajectories of points or objects from image sequences
  Main computational tools: dynamic filters (eg Kalman)

Motion analysis

What we will do

In summary

- Motion segmentation: change detection
- Correspondence: optical flow estimation
- Feature (corner) tracking

UniGe | MaLGa

Motion segmentation: change detection

Motion segmentation

Change detection

I_t

M_t

In the case the camera is still motion segmentation is can be implemented as a difference

Assuming we have a **reference image** I_{ref}

$$M_t(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - I_{ref}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

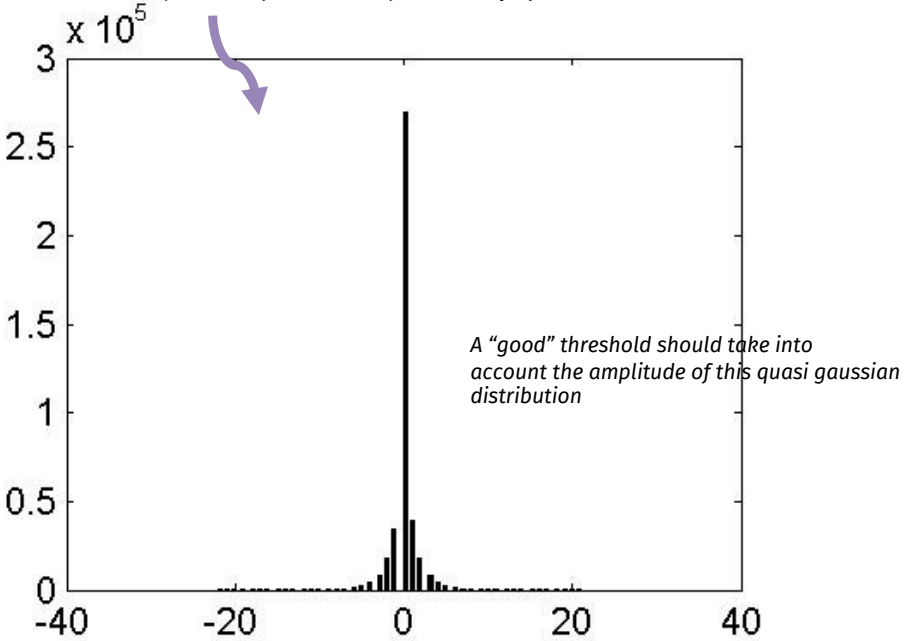
the threshold τ depends on the acquisition device and the acquisition conditions. It must be chosen considering a trade-off between false positives an false negatives.

The threshold

I_t

I_{t+1}

Histogram of the difference between 2 apparently identical images (two adjacent frames of an empty scene)



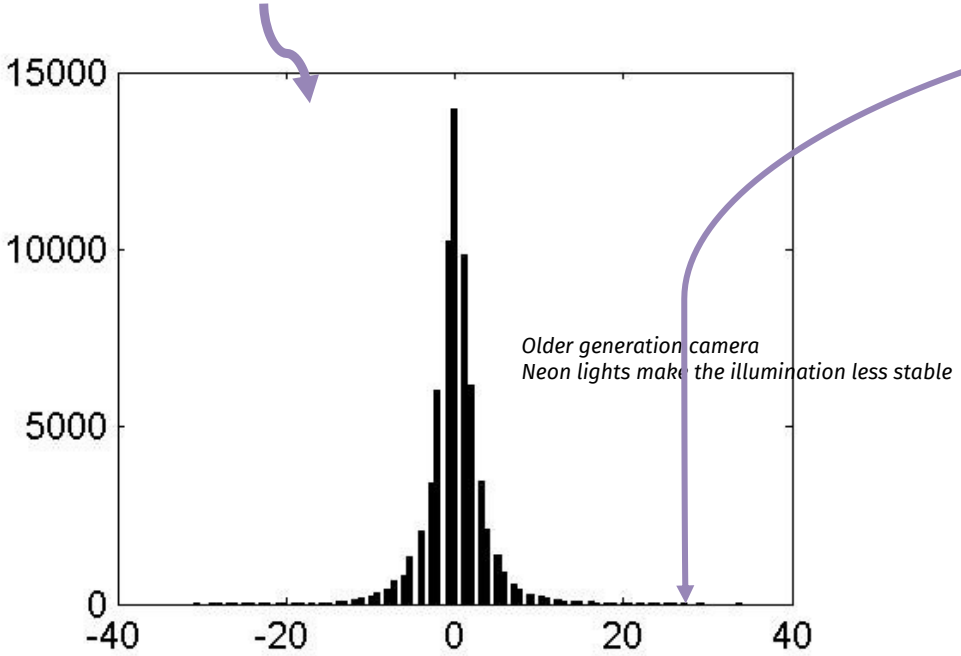
threshold=10

The threshold

I_t

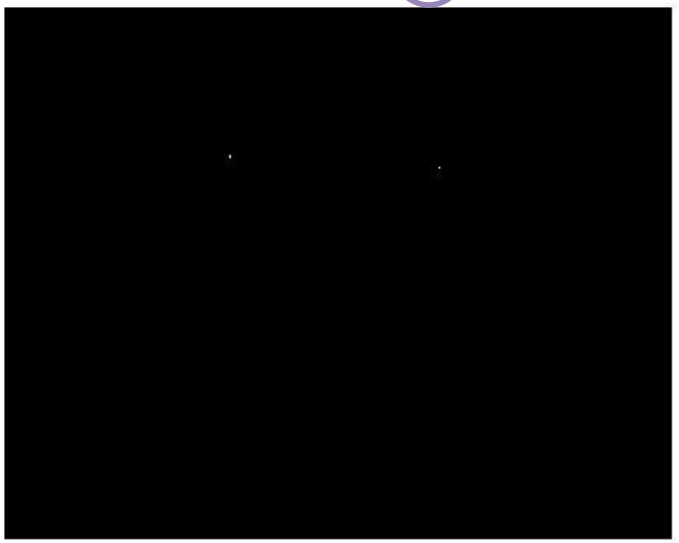
I_{t+1}

Histogram of the difference between 2 apparently identical images
(two adjacent frames of an empty scene)



Older generation camera
Neon lights make the illumination less stable

threshold=30



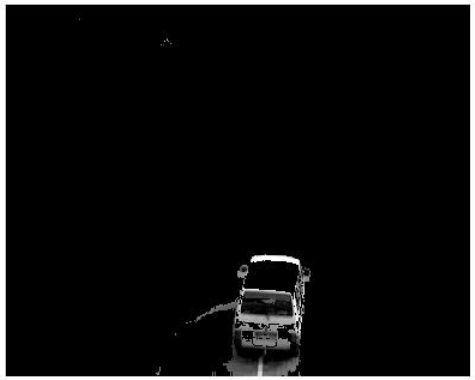
The reference image

the simplest way of detecting moving objects is to compare consecutive (or close) frames

frame 1

frame 2

|I1-I2|>th



In general it is not a good choice

Artifacts
Difficult to detect slow motion
Noisy localization

The reference image

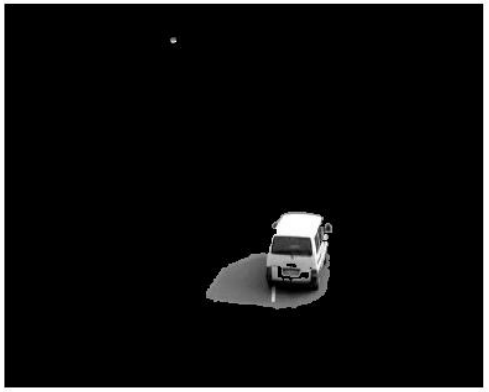
The reference image or, more in general the *background model*, is a “picture” of the empty scene, containing all the parts which are not moving

background

frame

|I-B|>th

after a cleaning procedure



change detection

The reference image – version 1

The reference image is a “picture” of the empty scene, containing all the parts which are not moving

The simplest way of computing it is to average the first N frames (assuming that at the beginning the scene is stationary)

$$I_{ref} = B = \frac{1}{N} \sum_{t=1}^{N} I_t$$

Change detection algorithm

Version 1

INPUT – image sequence $\{I_0,\dots,I_T\}$

Initialize I_{ref} with the first N images

$$I_{ref} = B = \frac{1}{N}\sum_{t=1}^{N} I_t$$

For each new frame $I_t \in \{I_{N+1}\dots I_T\}$

detect changes:

$$M_t(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - I_{ref}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

OUTPUT – binary maps sequence $\{M_{N+1},\dots,M_T\}$

PROS
- Very simple

CONS
- It assumes the background will not change in all the observation span (almost impossible in practice!)

Look at the door!

Change detection algorithm

Version 2 – running average

NOTICE : this method deals with *slowly changing* background (e.g, illumination changing in the scene)

INPUT – image sequence $\{I_0,\ldots,I_T\}$

Initialize the background B_0 as in version 1

For each new frame $I_t \in \{I_{N+1}\ldots I_T\}$

1. detect changes

$$M_t(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - B_{t-1}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

2. Update the background

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y) & \text{if } |I_t(x,y) - B_{t-1}(x,y)| > \tau \\ (1-\alpha)B_{t-1}(x,y) + \alpha I_t(x,y) & \text{otherwise} \end{cases}$$

OUTPUT – binary maps sequence $\{M_{N+1},\ldots,M_T\}$

notice: the background model is updated **at each frame**

Change detection

The reference image – version 2 – running average

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y) & \text{if } |I_t(x,y) - B_{t-1}(x,y)| > \tau \\ (1-\alpha)B_{t-1}(x,y) + \alpha I_t(x,y) & \text{otherwise} \end{cases}$$

PROS

- it is quite robust to moving objects in the scene
- it incorporates smooth stable changes (at a speed which is proportional to α)
- it is simple and computationally efficient

CONS

- it does not deal with repetitive (and uninteresting) motion

Change detection algorithm

Version … 3 – running average

With this version we try to deal with abrupt persistent variations (objects moved, a blade of light that enters, …)

INPUT – image sequence $\{I_0,\ldots,I_T\}$

Initialize the background B_0 as in version 1

For each new frame $I_t \in \{I_{N+1}\ldots I_T\}$

1. detect changes

$$M_t(x,y) = \begin{cases} 1 & \text{if } |I_t(x,y) - B_{t-1}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases}$$

2. Update the background

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y) & \text{if } |I_t(x,y) - I_{t-\Delta t}(x,y)| > \tau' \\ (1-\alpha)B_{t-1}(x,y) + \alpha I_t(x,y) & \text{otherwise} \end{cases}$$

"a few frames back"

OUTPUT – binary maps sequence $\{M_{N+1},\ldots,M_T\}$

notice: the background model is updated **at each frame**

Motion segmentation

General case

If the camera is moving, we need to consider the presence of a dominant motion (the ego motion)

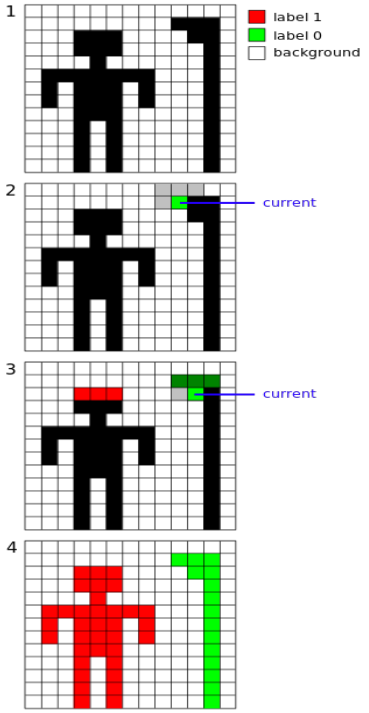
A possible approach is to first obtain a rough estimate of the ego motion and then compensate it to generate a synthetic datum *as if* the camera were still

Much harder and computationally intensive (as an alternative we often look for semantic info: people detection, car detection, ...)

blob tracking

Blob detection

- Mt is a binary map where all pixels are still individual entities
- To obtain larger ”entities” we compute connected components (objects or parts of objects) which we normally refer to as “blobs”
- Connected components are image segments were each pair of pixel can be connected by a path remaining inside the segment



from Wikipedia

Blob descriptors

- Position
- velocity (direction, magnitude)
- shape descriptors (area, perimeter, aspect ratio, higher order moments)
- texture, color, ...

Blob tracking

The goal of tracking is to associate each blob at time t+1 with one of the blobs observed in previous instants {0,...,t}

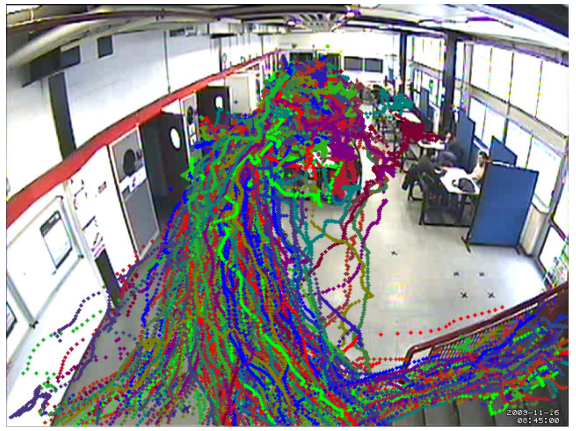
The two main ingredients are

- similarity between blobs (in adjacent time instants): we may extend what we know on features matching
- temporal continuity (dealing with temporary gaps and integrating observations along longer time frames). This is related to dynamic filtering

UniGe | MaLGa

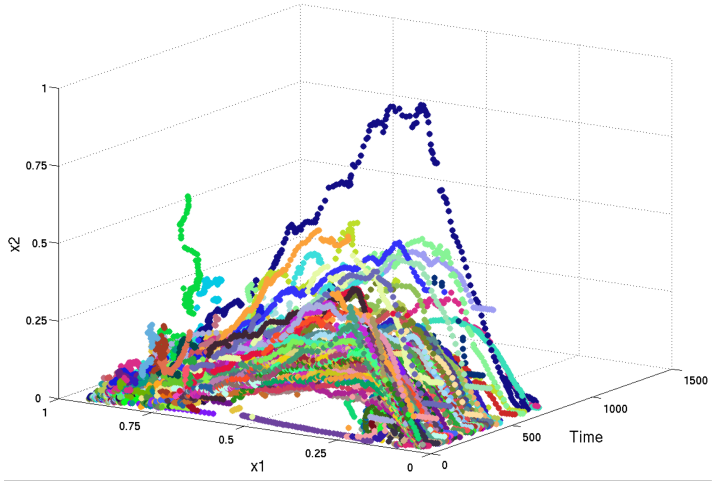
from tracking to temporal sequences

After a few hours we may
Gather quite a lot of measurements



for each moving object in the scene we may collect a **temporal sequence** of instantaneous observations

$$\{\mathbf{x}_i\}_{i=1}^{N} \quad \text{with} \quad \mathbf{x}_i = (x_i^1, x_i^2, \ldots, x_i^{k_i})^\top$$



instantaneous observation (may include different types of information, such as position, shape, speed, …)

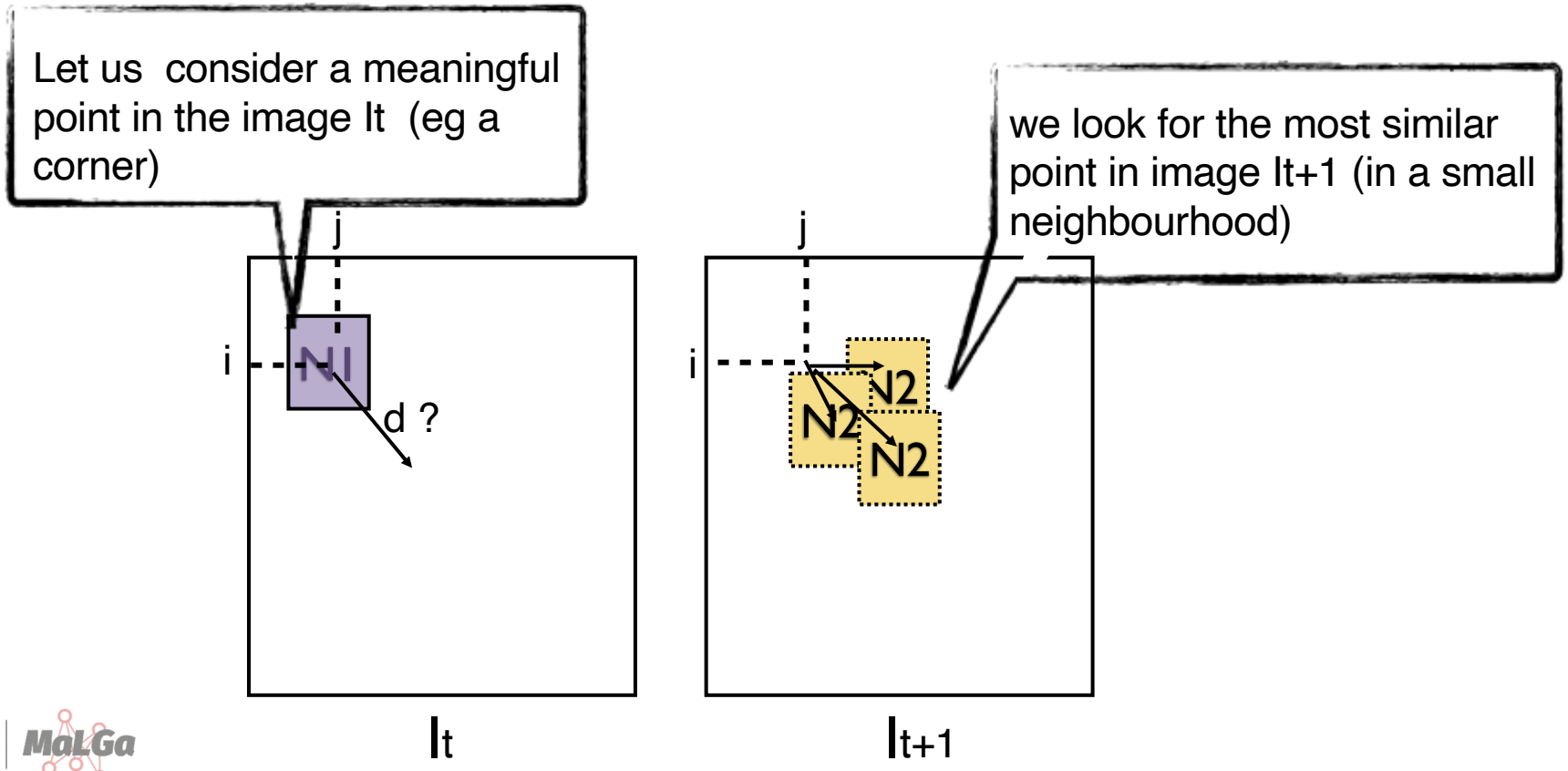
UniGe | MaLGa

Correspondence: optical flow

Motion analysis as a correspondence problem

correspondence: which elements of a frame correspond to which elements of the next frame?

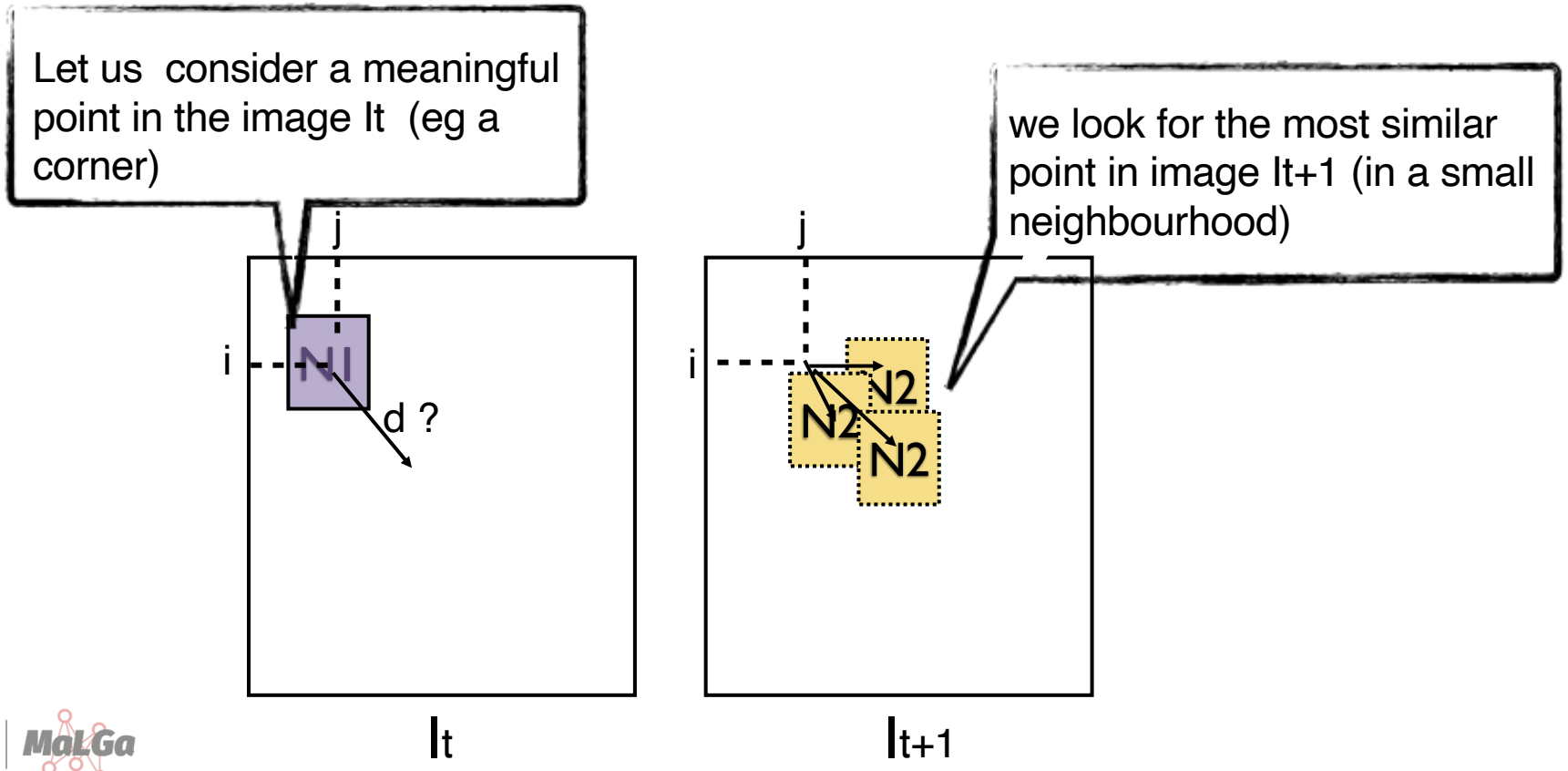
We take advantage of the high similarity between adjacent frames



Motion analysis as a correspondence problem

We take advantage of the high similarity between adjacent frames

1. Correlation-based approaches
2. Gradient-based approaches



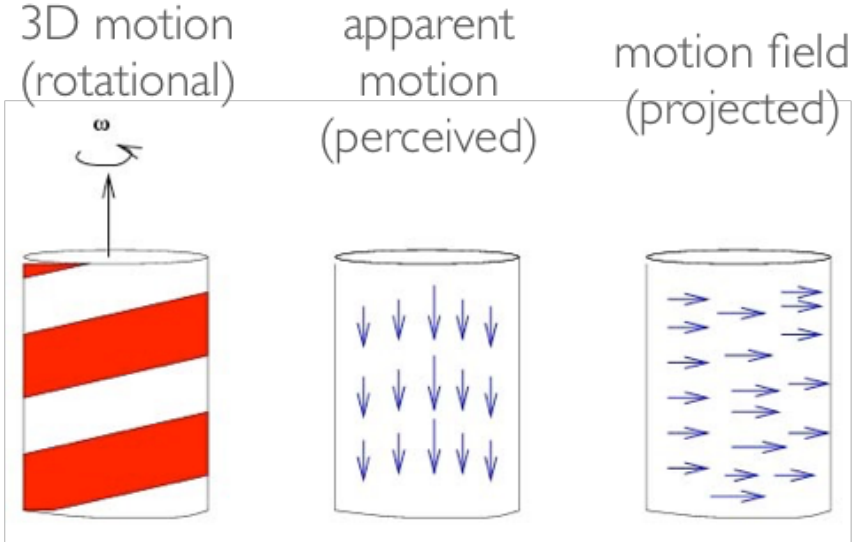
Motion analysis as a correspondence problem

In the case of motion, correspondence may also be seen as a problem of *estimating the apparent motion of the brightness pattern* (the so called **optical flow**)

Why ”apparent”? *Because it’s not the real motion, but the one we perceive*

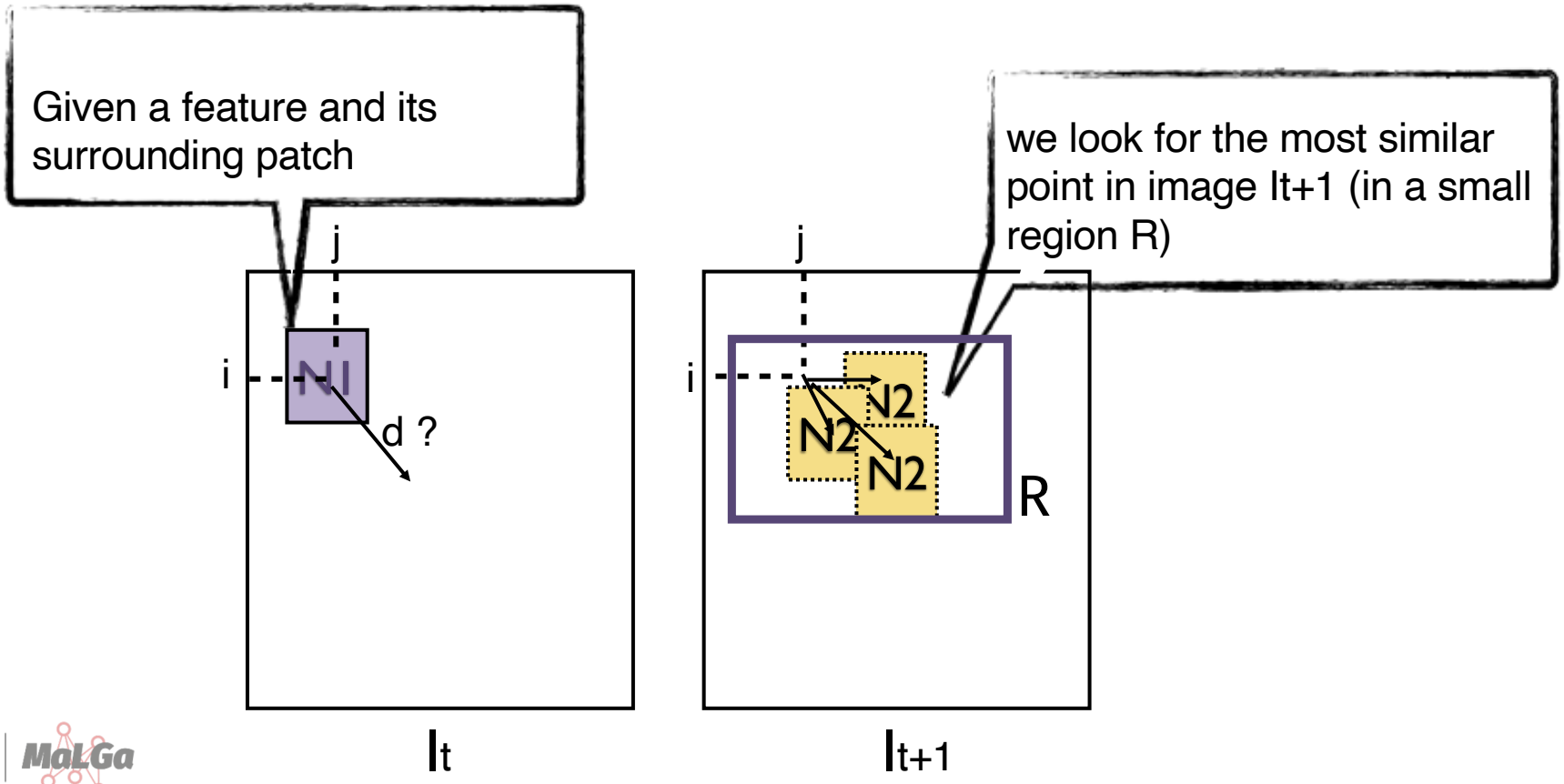
Example1. A ball is bouncing in a dark room: what do we perceive? Nothing!

Example2: the barber pole



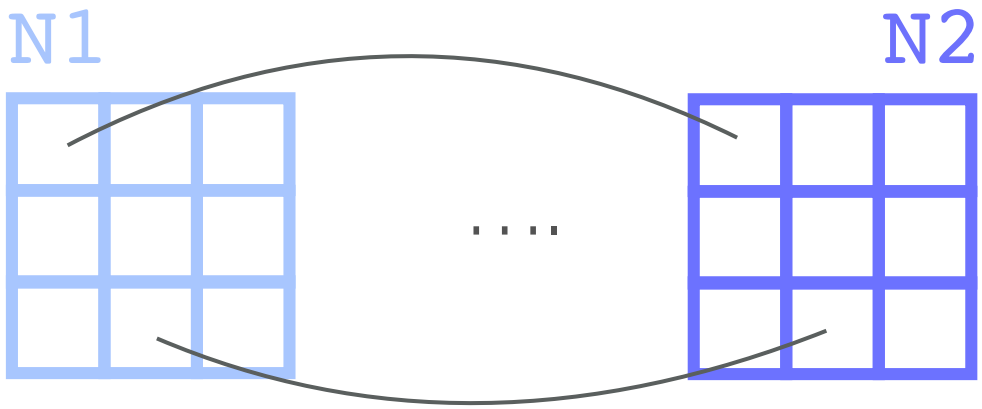
Motion analysis as a correspondence problem

Correlation-based approaches



Correlation-based approaches

We can measure similarity between pairs of patches in a pixel-wise manner



$$\phi(N1, N2)$$

Similarity measures for image patches

SUM OF SQUARED DIFFERENCES

$$\phi_{SSD}(N1, N2) = - \sum_{k,l=-\frac{W}{2}}^{\frac{W}{2}} (N1(k,l) - N2(k,l))^2$$

NORMALIZED CROSS CORRELATION

$$\phi_{NCC}(N1, N2) = - \sum_{k,l=-\frac{W}{2}}^{\frac{W}{2}} \frac{(N1(k,l) - \mu_1)(N2(k,l) - \mu_2)}{W^2 \sigma_1 \sigma_2}$$

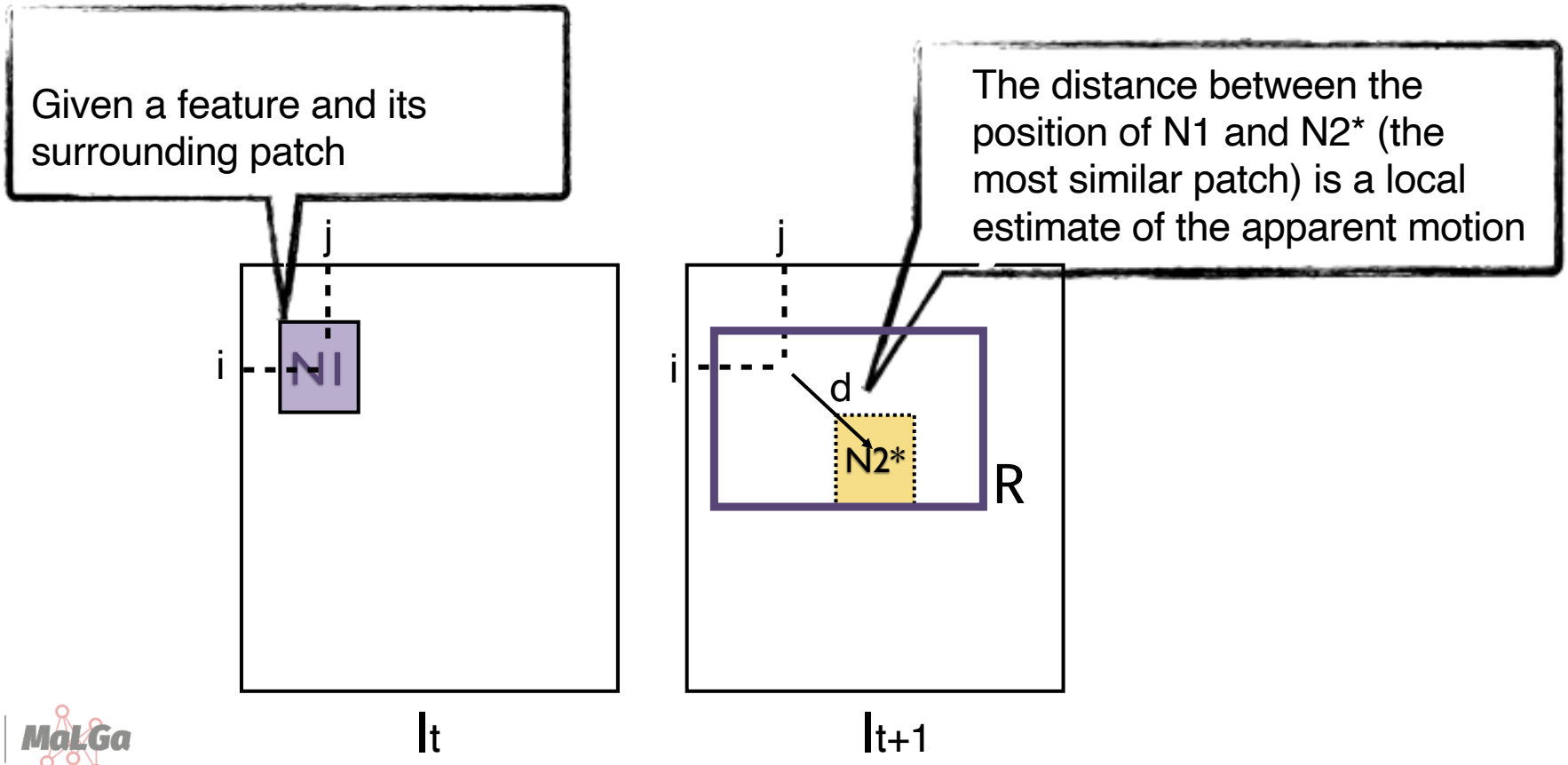
$$\mu_i = \frac{1}{W} \sum_{k,l=1}^{W} Ni(k,l)$$

$$\sigma_i = \sqrt{\frac{1}{W} \sum_{k,l=1}^{W} (Ni(k,l) - \mu_i)^2} \quad \text{with } i = 1, 2$$

NCC: values in the range [-1, 1]

Motion analysis as a correspondence problem

Correlation-based approaches



Motion analysis as a correspondence problem

Gradient-based derivation

It's based on the image brightness constancy equation: from one frame to the next the image appearance does not change

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

If the motion is small, we can use the following Taylor approximation

$$I(x + u, y + v, t + 1) = I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial x}v + \frac{\partial I}{\partial t} \quad + H.O$$

We thus obtain $\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$

Or more compactly

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

$$(\nabla I)^\top \mathbf{u} + I_t = 0$$

Optical Flow

Gradient-based derivation

The optical flow is a vector field subject to the constraint

$$(\nabla I)^\top \mathbf{u} + I_t = 0$$

This constaint gives us 1 equation per each image point

Notice that **one** constraint is not enough to compute the optical flow (2 unknowns)

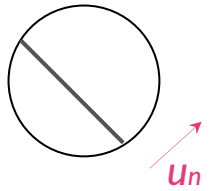
Optical Flow

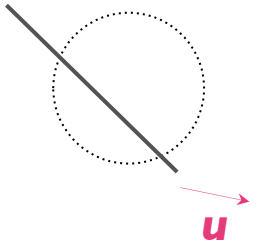
The aperture problem

the image brightness constancy equation allows us to determine the optical flow component parallel to the spatial image gradient (normal flow)

Analytically

$$u_n = \frac{(\nabla I)^\top \mathbf{u}}{||\nabla I||} = \frac{-I_t}{||\nabla I||}$$





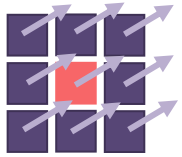
Optical Flow Algorithms

Lucas Kanade Algorithm

many algorithms start from the idea of adding constraints to the underdetermined system obtained by the brightness constancy equation

we will see a simple way of doing so: the Lucas-Kanade algorithm

- assumption: **u** is constant in a small neighbourhood of a point

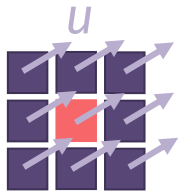


Optical Flow Algorithms

Lucas Kanade Algorithm

the assumption allows us to obtain a system of equations with one equation for each point in the neighbourhood

$$(\nabla I(\mathbf{x}_i, t))^\top \mathbf{u} + I_t(\mathbf{x}_i, t) = 0 \quad \mathbf{x}_i \in N$$



we then obtain a linear system **Au=b** with

$$A = \begin{bmatrix} \nabla I(\mathbf{x}_1, t)^\top \\ \nabla I(\mathbf{x}_2, t)^\top \\ \cdot \\ \cdot \\ \cdot \\ \nabla I(\mathbf{x}_m, t)^\top \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} -I_t(\mathbf{x}_1, t) \\ -I_t(\mathbf{x}_2, t) \\ \cdot \\ \cdot \\ \cdot \\ -I_t(\mathbf{x}_m, t) \end{bmatrix}$$

m elements in the N neighbour.

Optical Flow Algorithms

Lucas Kanade Algorithm

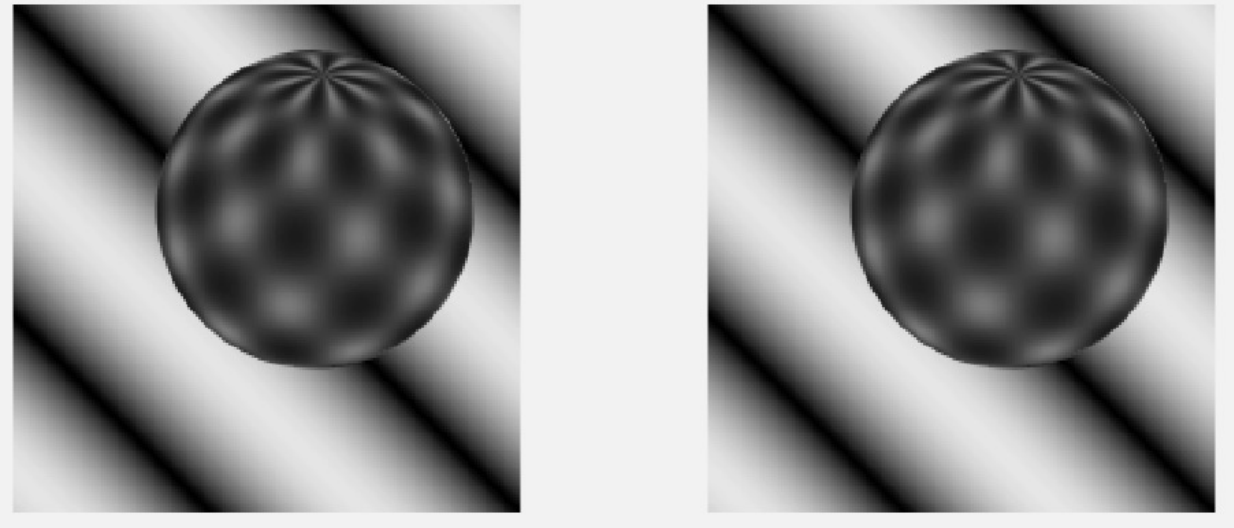
The linear system may be solved with the pseudo-inverse

$$\mathbf{u} = A^{\dagger}\mathbf{b} \quad \text{with} \quad A^{\dagger} = (A^{\top}A)^{-1}A^{\top}$$

notice that the inversion of the matrix will be ill-posed if the matrix is not full rank

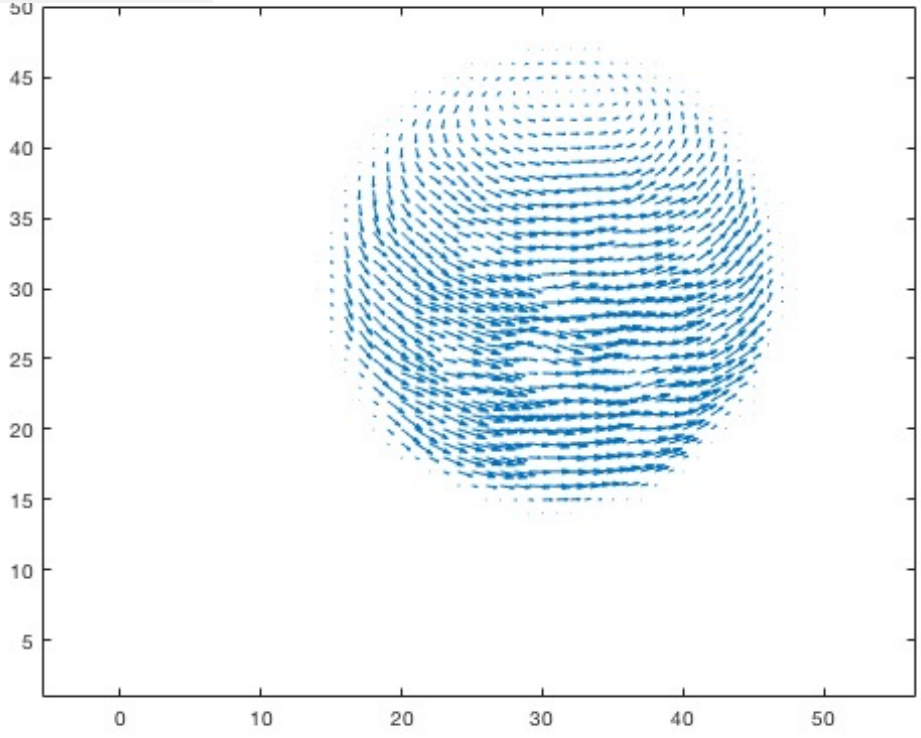
The matrix is full rank in the proximity of corners (points who do not suffer from the aperture problem)
This is why often times Lucas Kanade is implemented as a *sparse* algorithm (after a corner detection stage) – see for instance OpenCV implementation

Lucas Kanade example

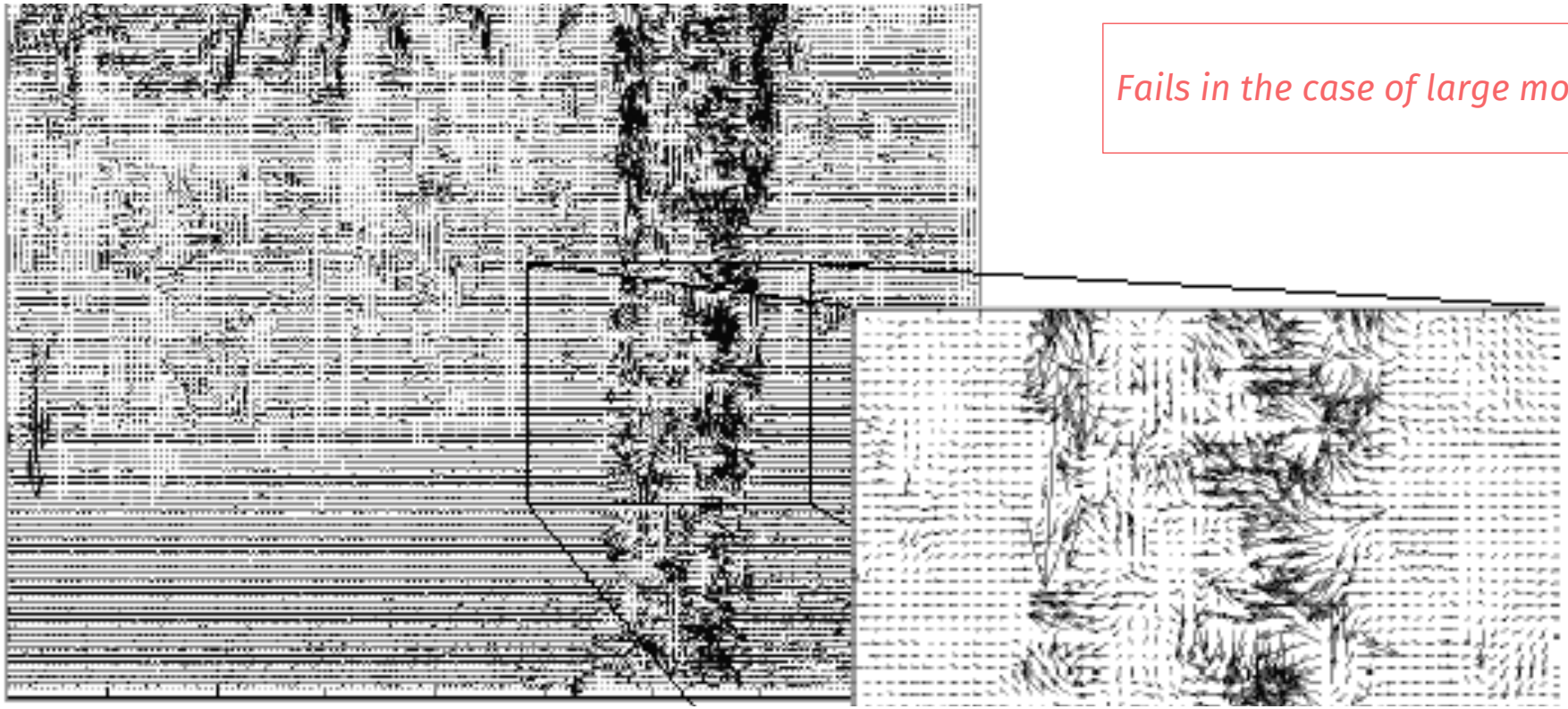


Lucas Kanade on a synthetic highly textured sequence of a *slowly* rotating sphere

Small displacements!

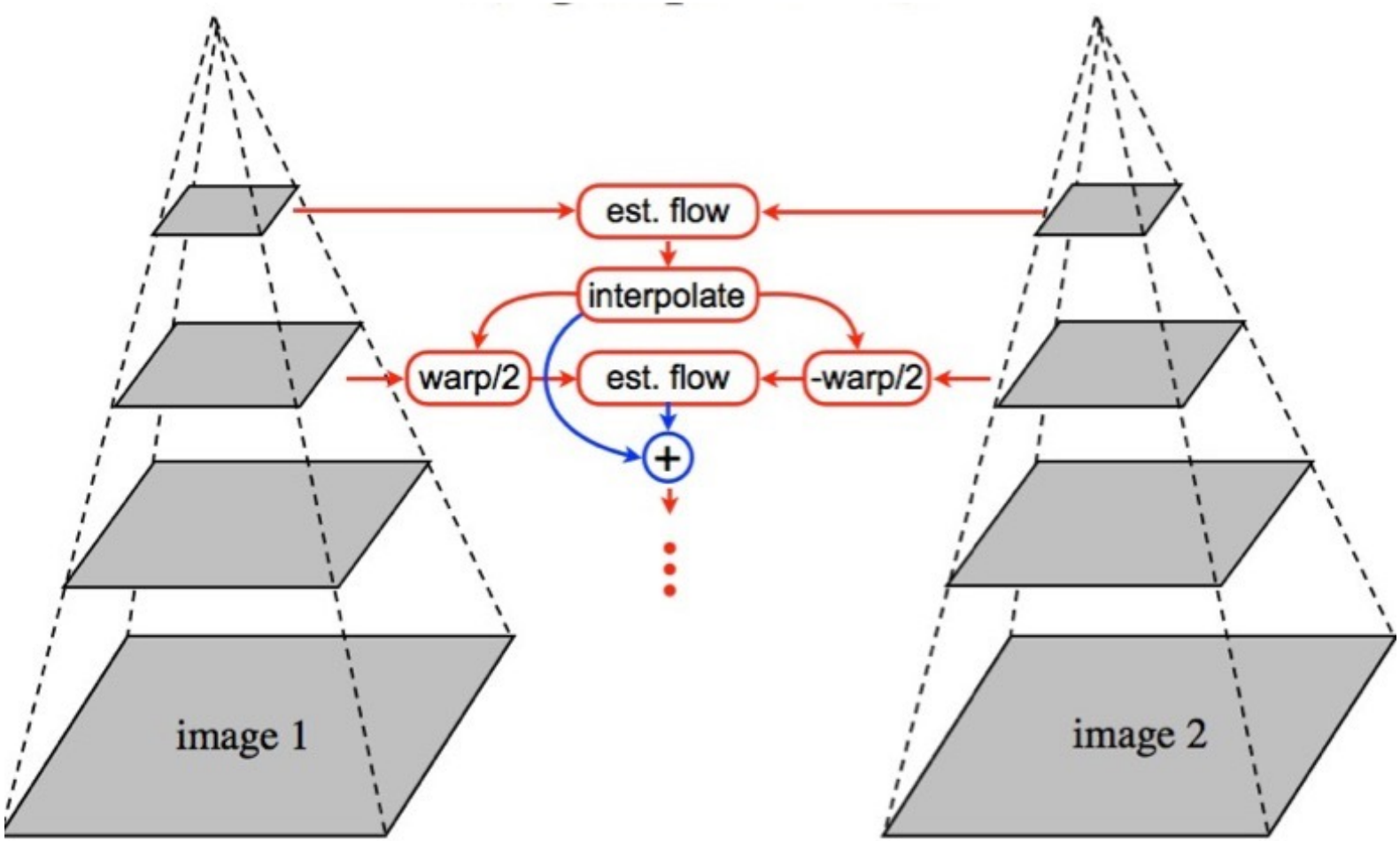


Lucas Kanade example



Fails in the case of large motion

coarse to fine estimation



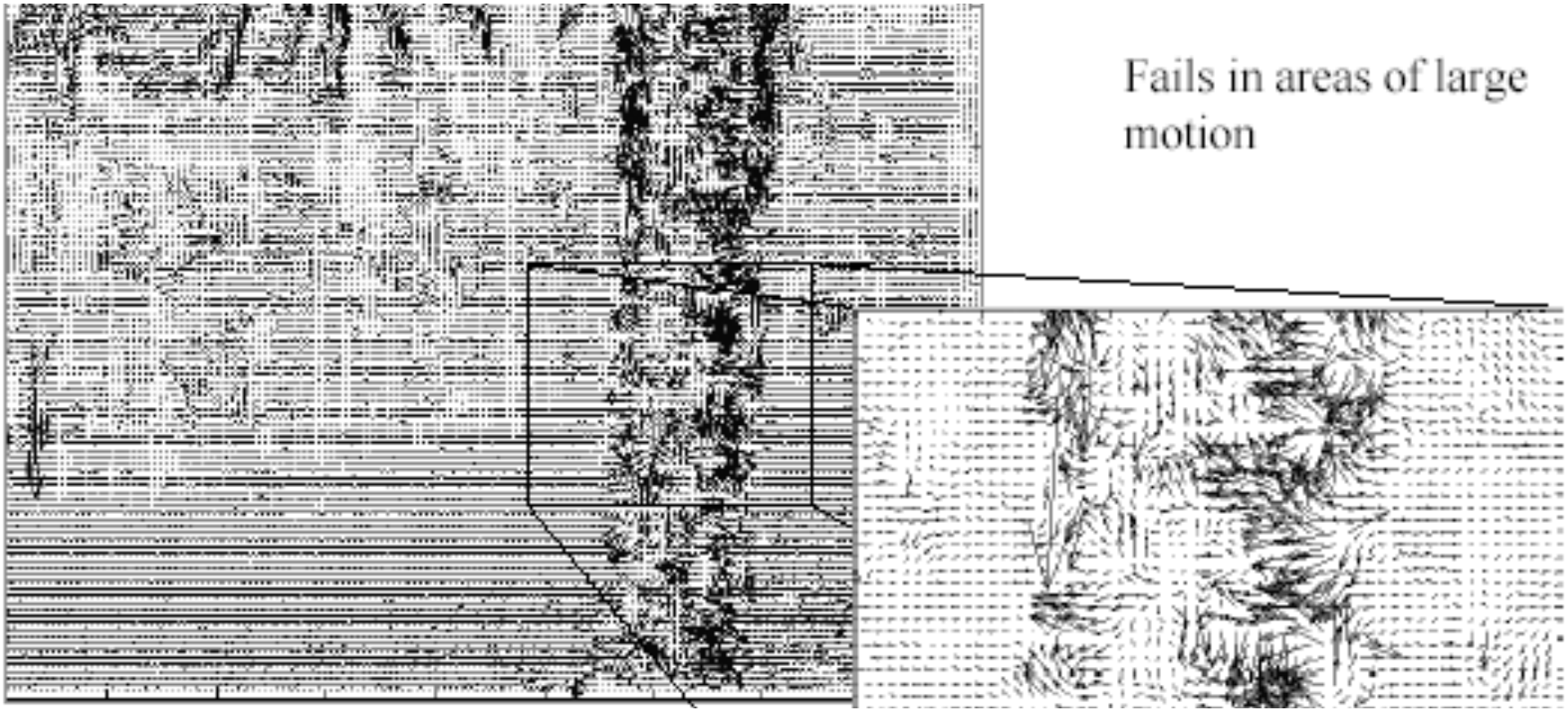
The image frames are first represented as Gaussian Pyramids (see last week's class)

Coarse to fine estimation algorithm

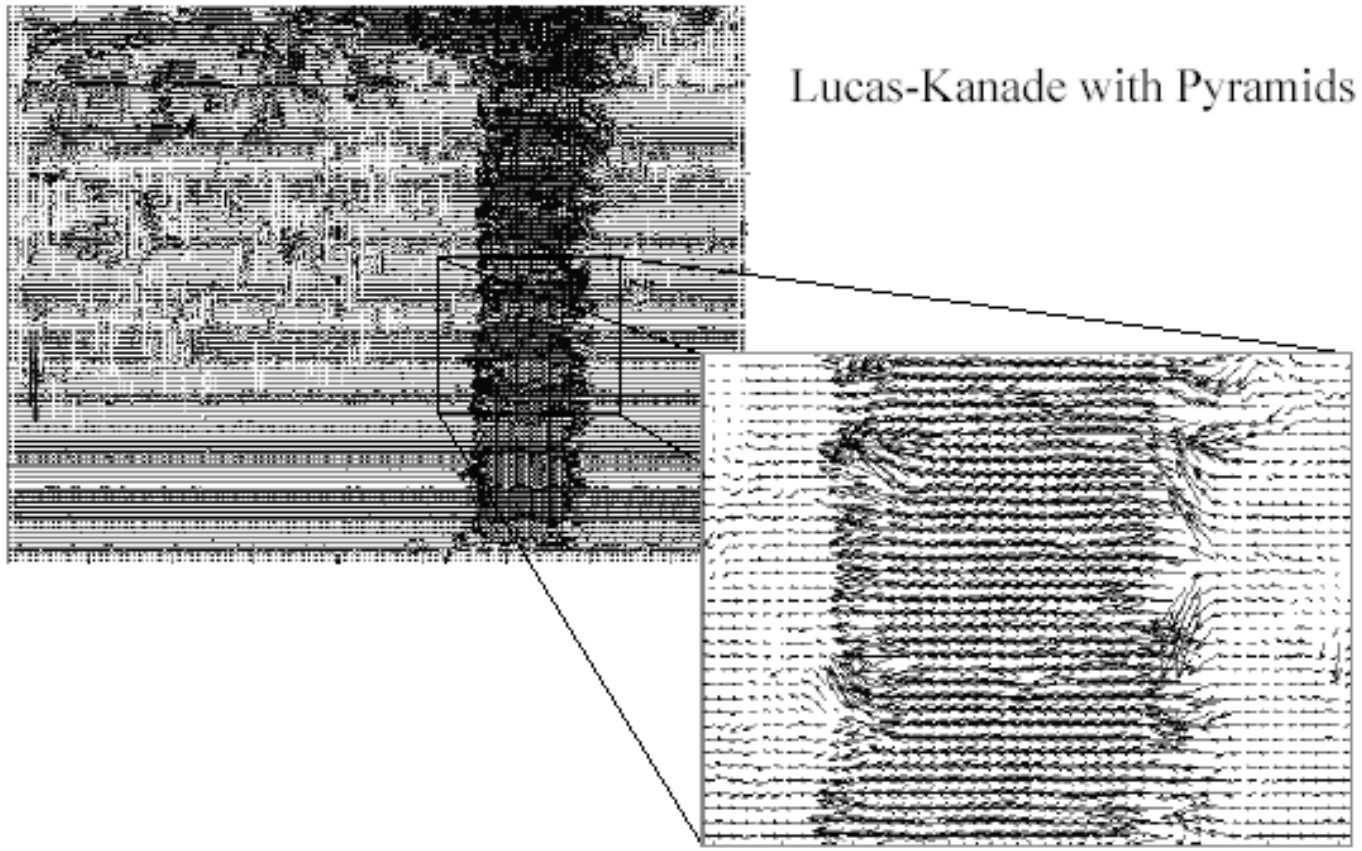
At each level i

- Take flow estimated in the previous level: u(i-1)
- Upsample the flow and create a first estimate of u(i)
  (double the size of the matrices and multiply the values by 2)
- Warp I(t) w.r.t u(i) and apply LK to Iwarp(t) and I(t+1) obtaining a correction u'(i)
- Update u(i) by adding u'(i)

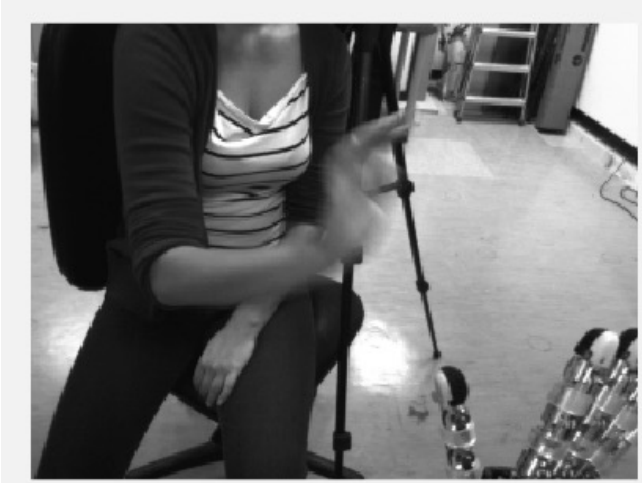
Lucas Kanade example



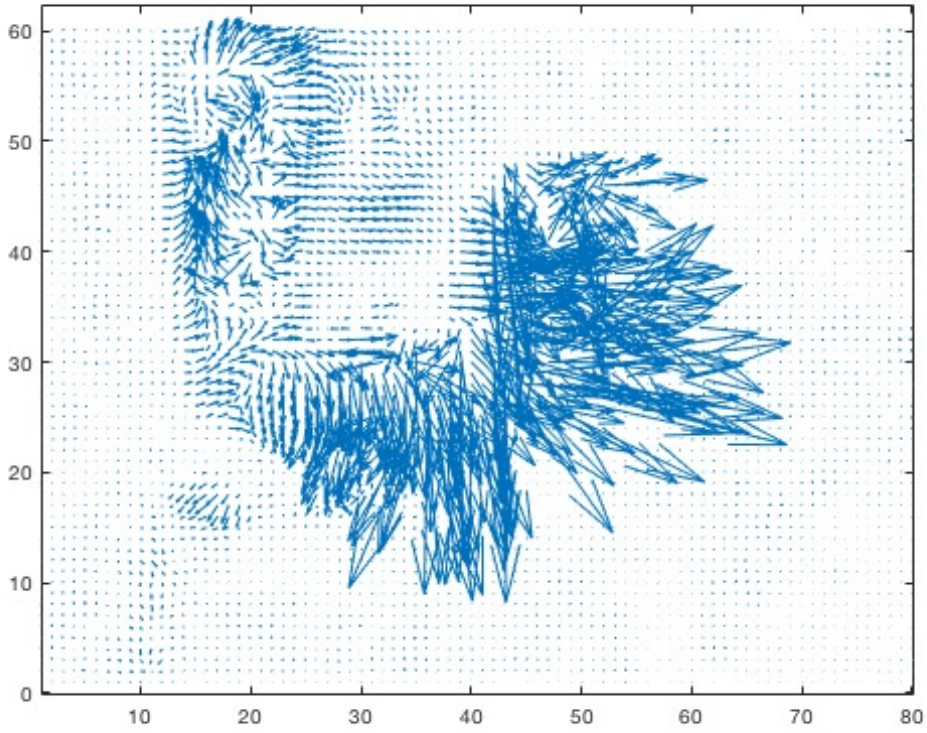
Multi-resolution Lucas Kanade example



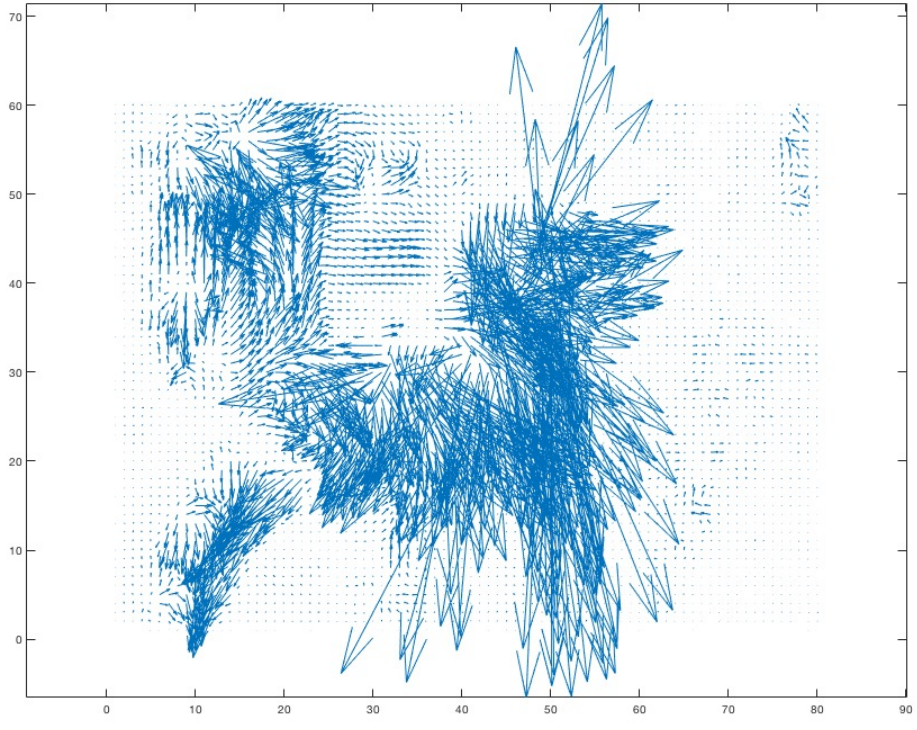
lucas kanade another example (less texture)



Lucas Kanade

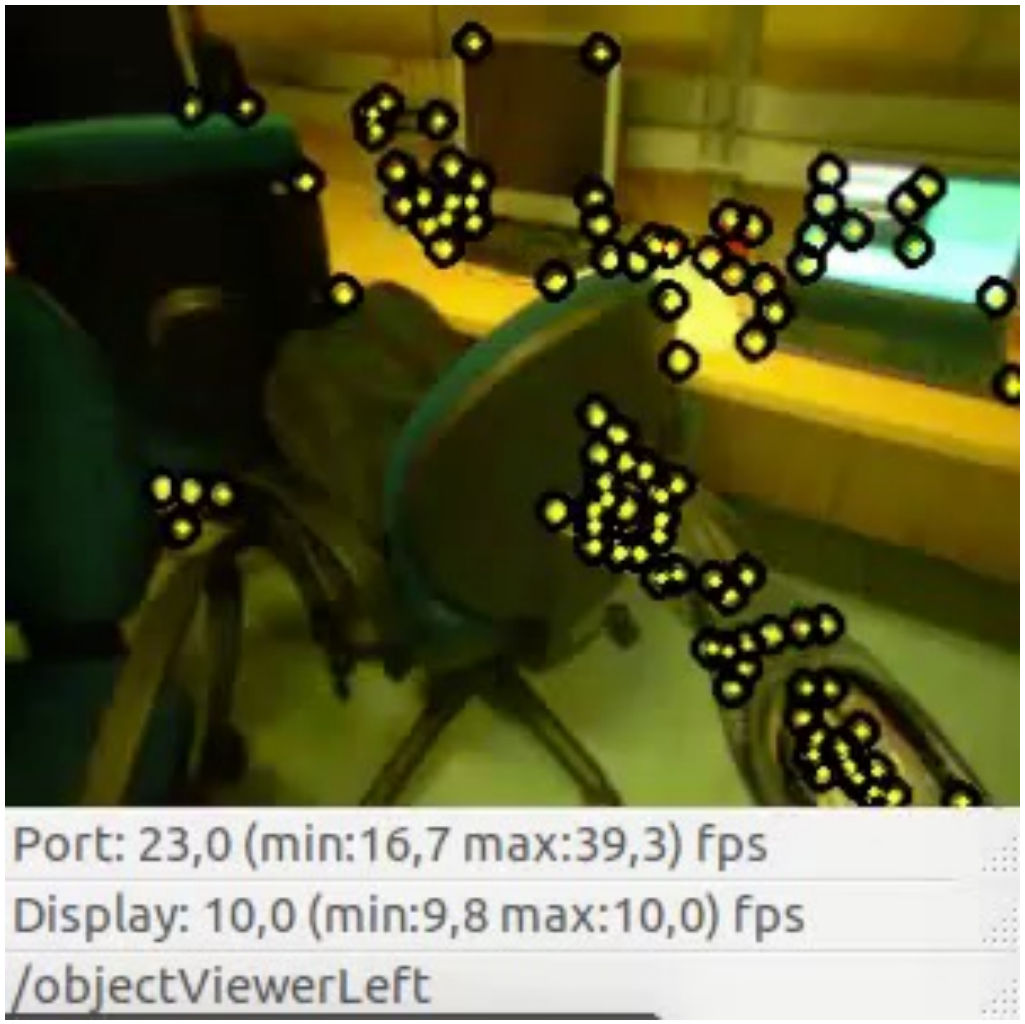


Lucas Kanade hierarchical version

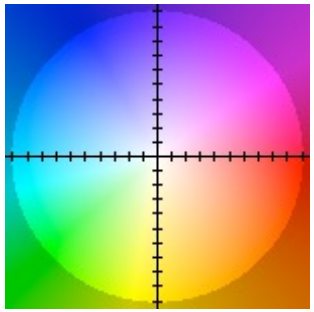


lucas kanade - sparse implementation (opencv)

Optical flow estimated only on points not suffering from the aperture problem (corners)



example - dense estimates



these results have been obtained with the **Färneback method** available in OpenCv

this method applies a more complex parametrization (polynomial) over the pixel neighbourhood, but still maintain a very high efficiency.

UniGe | MaLGa

Correspondence over time: feature tracking

Feature tracking: multi-frames algorithm

input:

- a video sequence I(0), I(1), ..., I(T)

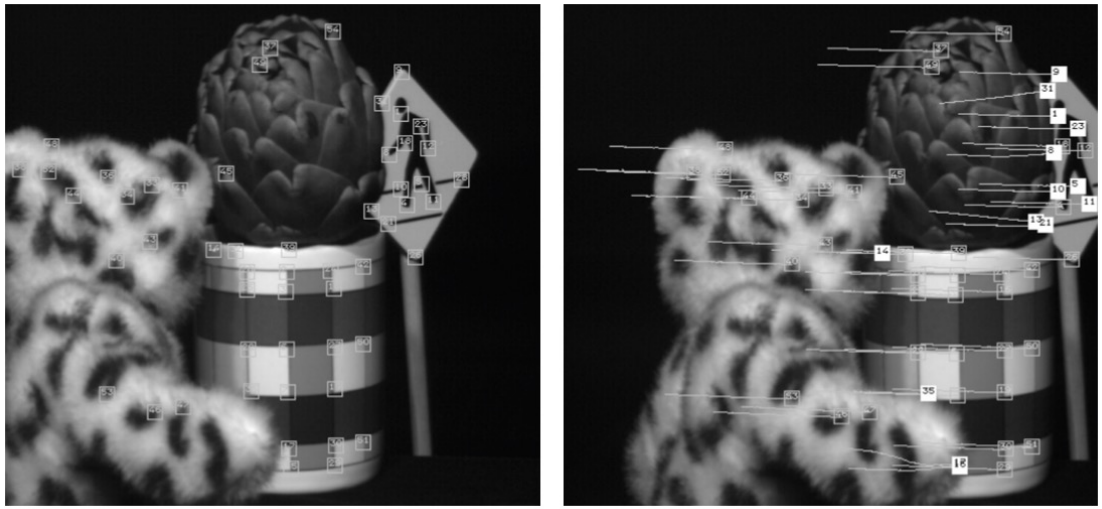
Here we only extract keypoints on the first framev(we could add subsequent extractions)

Extract corners from frame I(0): c_i with i=1,...,N

for each pair of adjacent frames I(t) I(t+1)

- for each corner c_i(t) on I(t)
  - Estimate it optical flow and obtain an updated position c_i(t+1)
  - Update the c_i track

Output: N corner tracks

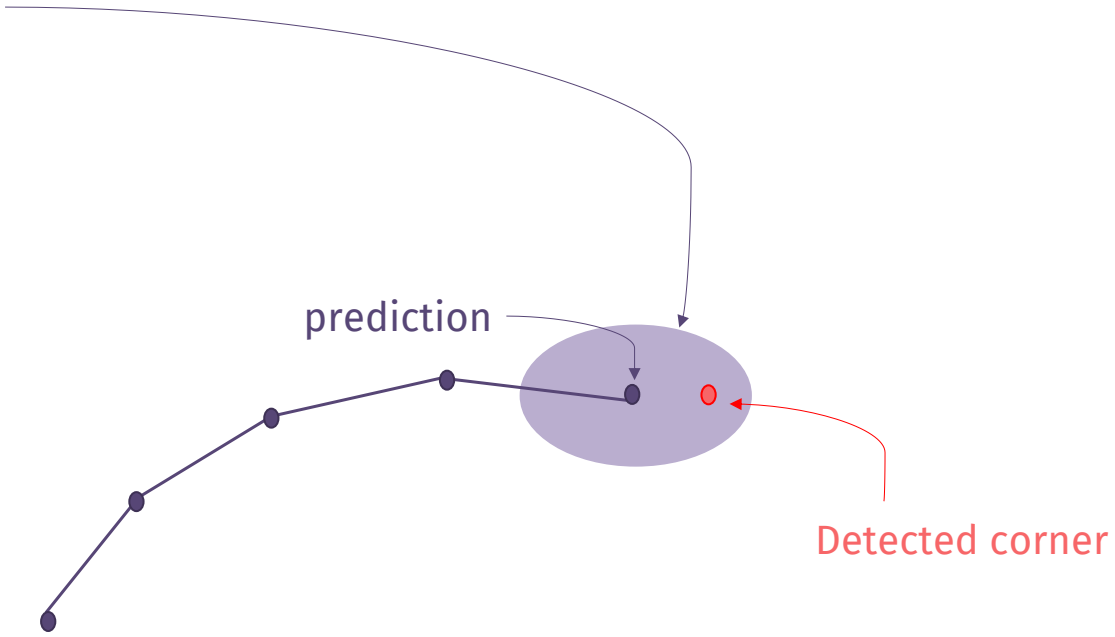


Feature tracking: the role of Kalman filter

Kalman filter allows us to smooth the noisy trajectories

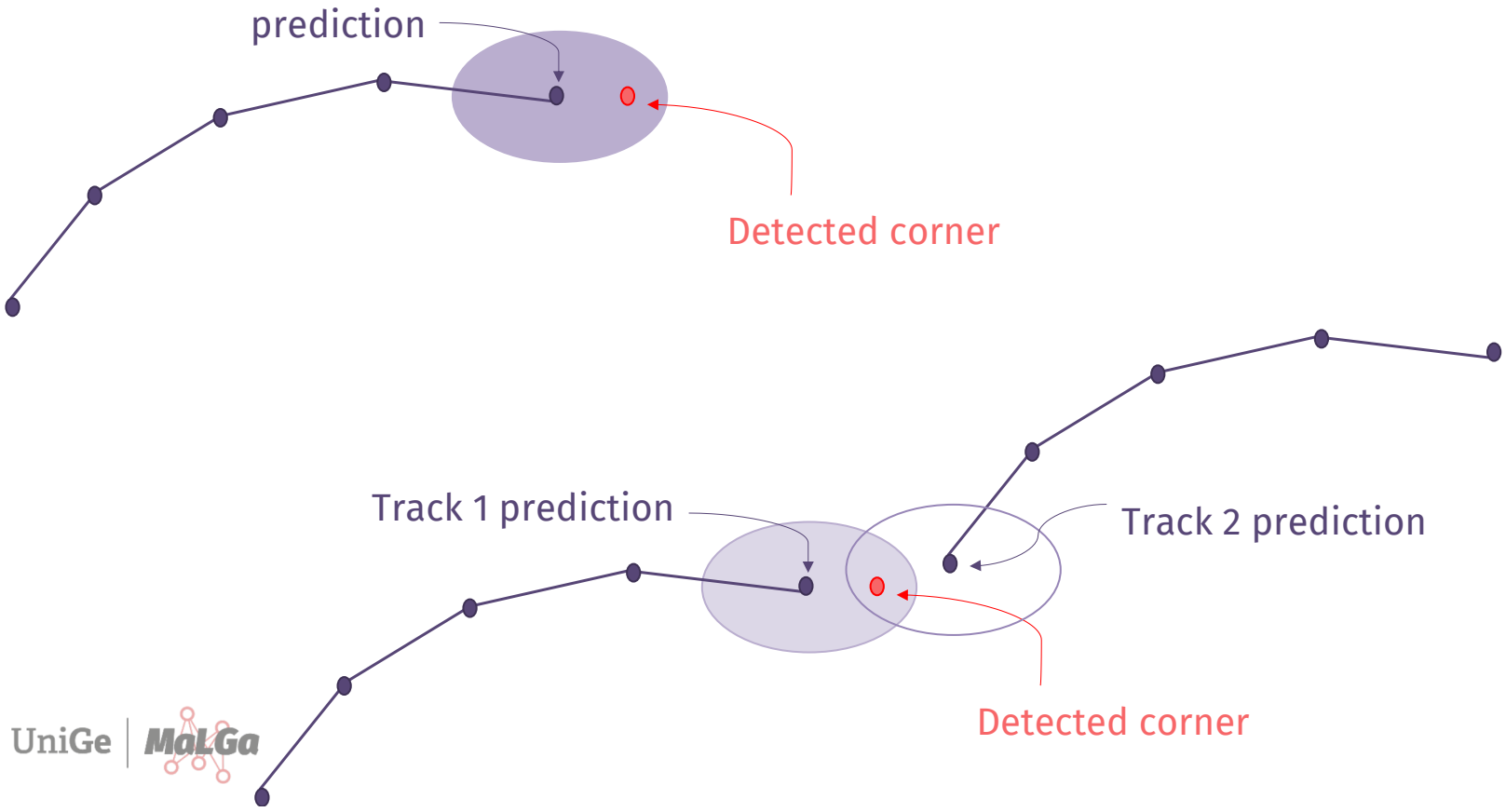
Further, it can be used to fill trajectory gaps: we may use the estimated state as a hypothesis of the missing measurement

The P_t state covariance matrix provides us with a measure of the state uncertainty ellipse



Feature tracking: data association problem

When we are tracking multiple corners, we often face data association problems



UniGe