



---

# TSBB15

# Computer Vision

## Lecture 7

## Object Tracking Project



# Lab 1

---

Not everyone finished lab1 yesterday.

Opportunity to show the completed lab:  
Tue 11/2 9.00 in Olympen



# Project goals

## Technical goals of this project

Given an image sequence,  
**detect** and **track** all  
moving objects

Indicate objects with  
**boxes** + **identity numbers**

## Assumptions

Static camera

Static background

Humans or cars are moving  
on a flat ground plane



PETS09-S2L1



# Project goals

## Technical goals of this project

Given an image sequence,  
**detect** and **track** all  
moving objects

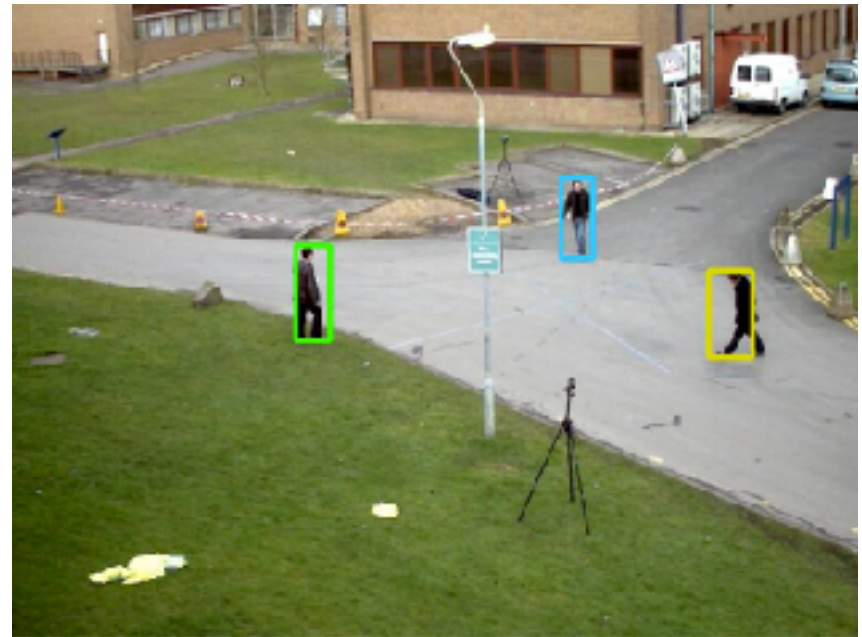
Indicate objects with  
**boxes** + **identity numbers**

## Assumptions

Static camera

Static background

Humans or cars are moving  
on a flat ground plane



PETS09-S2L1-GT



# Object tracking

---

## Application areas

surveillance, traffic analysis, sports, ...

## Commercial systems

[Axis](#), Chyron Hego ([Tracab](#)), [Irisity](#), [Viscando](#)

## Research projects

[iPatch](#) (pirate detection at sea)

[VOT](#) (visual object tracking challenge)

[PETS](#) (annual workshop)

## Student projects in TSBB11



# Project web page

---

Full specification of the project on webpage:

[www.cvl.isy.liu.se/education/undergraduate/tsbb15/object-tracking-project](http://www.cvl.isy.liu.se/education/undergraduate/tsbb15/object-tracking-project)

Requirements

Datasets

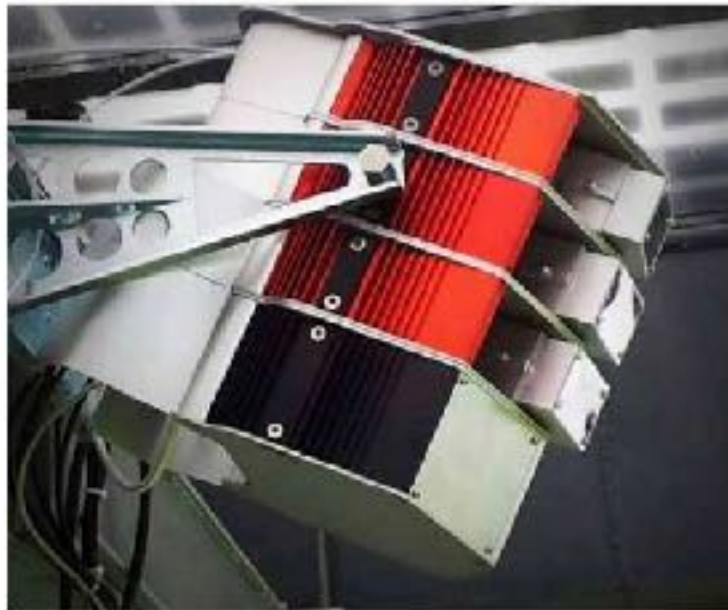
References

...



# Sports Analysis

Example:  
Tracab system (from company  
ChyronHego) for automatic tracking of  
football players.



3D tracking in real time  
using a pack of stereo rigs.



# Project Challenges

---

## Shadows / reflections from moving objects

Create “ghost” objects

## Uneven illumination of the scene

Visual appearance of an object changes

Variation both in space and in time

## 3D rotations

Object appearance changes

## Occlusion

Ambiguities in object identity





# Toolbox of methods

---

Since we have a static camera & background:

Use *background modeling* to detect motion

To robustly maintain object identity:

KL-tracking, Foreground Model, Motion Prediction,  
Ground plane mapping + 3D tracking

Shadows/reflections can be managed by:

Colour modeling



# Background modeling

---

A popular application of *mixture models* [LE6]

- A background model is a set of mixture models; one for each pixel in the image
  - 1 component is sometimes sufficient
  - 2 or more components to handle more general scenes
- Pixel values far from the mixture model are seen as foreground pixels = moving objects
- Popular way to track e.g. people and cars in stationary surveillance cameras
- Fast compared to dense motion estimation



# Background modeling

- Master thesis project at CVL, LiU, by John Wood
- Background modeling + shadow detection





# Background modeling

---

Two proposed approaches:

- **Gaussian Mixture Models [LE6]**

Pseudo-code for on-line updates in J. Wood's master thesis, section 4.3

- **Median filtering**

Masters thesis of J. Wood, section 2.2

PhD thesis of H. Ardö, section 3.2.2

Simple and thus useful to start with.



# Foreground segmentation

---

Background modeling gives a **likelihood map** with  $p(I(x,y)|\text{background})$  [See LE6]

Use likelihood map to find **consistent regions**  
thresholding (preceded by filtering for cleanup)  
morphological shrink + expand for cleanup  
labeling

From these regions, extract **bounding boxes**  
The smallest rectangle that includes the detected object



# Maintaining object identity

---

Start by detecting moving objects in each frame

Next: determine which detection in frame  $t$   
corresponds to which one in frame  $t + 1$



# Maintaining object identity

---

Start by detecting moving objects in each frame

Next: determine which detection in frame  $t$  corresponds to which one in frame  $t + 1$

Simple approach:

Check how much all pairs of detections overlap

In general this leads to the **association problem**

Find the optimal assignment of correspondence between detections in frame  $t$ , and  $t+1$ , given **match scores**.

Solutions: Hungarian methods

In this project, greedy approximations are acceptable.



# Maintaining object identity

Example: overlap scores

Frame t+1 regions: 0                    1                    2                    3                    4

Frame t regions:

0	0.6	0.0	0.0	0.0	0.0
1	0.0	0.5	0.55	0.0	0.0
2	0.0	0.0	0.0	0.87	0.0
3	0.0	0.0	0.0	0.0	0.62





# Maintaining object identity

Example: overlap scores, assignment

Frame t+1 regions: 0                    1                    2                    3                    4

Frame t regions:

0	0.6	0.0	0.0	0.0	0.0
1	0.0	0.5	0.55	0.0	0.0
2	0.0	0.0	0.0	0.87	0.0
3	0.0	0.0	0.0	0.0	0.62



# Maintaining object identity

---

Some possible **match scores** for assignment:

Overlap of bounding box

Overlap of *predicted* bounding box

Overlap of region mask

Overlap of warped/*predicted* region mask

Overlap + visual appearance score

etc...



# Temporal prediction

---

## Prediction of bounding box/mask locations

From temporal filtering we can predict where the object will end up in the next frame

e.g. IIR, Kalman, KLT tracks, optical flow

## Can help to maintain object identity in case of

occlusion

close objects, rapid motion

or noise in general



# Ground plane modeling

## Assumption:

objects are moving on a planar surface:  
the ground plane.

Mapping from image coordinates to  
ground plane is a homography, which  
can be estimated (TSBB06).



- “foot points” of object can be mapped to a position in the ground plane
- The ground plane position can be temporally filtered
- **Hand-over** to other views is possible



# Datasets

---

The scientific community has produced a number of standard datasets for *multiple object tracking*:

CAVIAR

PETS and MOT (annual competitions)

IVSS

[changedetection.net](http://changedetection.net)

They typically contain:

Video sequences (MPEG + JPEG)

Annotation (labelling of objects) in some datasets



# Performance evaluation

---

Compare your boxes with *ground truth* (GT)

Possible performance measures:

- Average overlap error

- Number of identity switches for an object

- ...

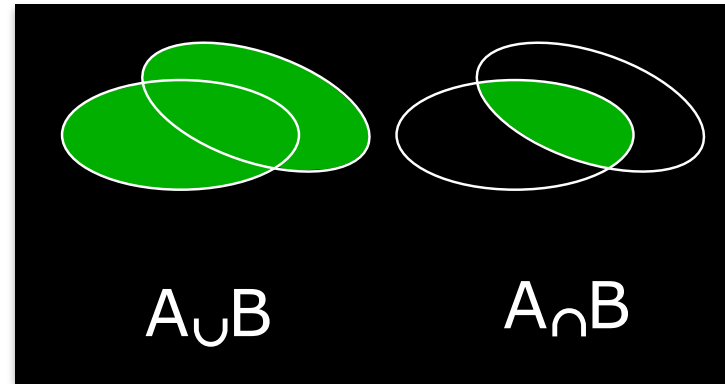
- [Details on project webpage]



# Overlap

Intersection over union (IoU)  
aka. Jaccard index:

$$J(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$$



Overlap error (aka. Jaccard distance):

$$\epsilon = 1 - \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$$



# Training set/test set

---

## Parameter tuning

Many system parameters to tune

Define a training set/test bench for parameter tuning

Test on new sequences - with the same parameters!

See discussion in learning lecture [LE6]

## Benchmark (test set) will be decided later

Last year it was three sequences from CAVIAR

We will select new sequences this year





# Automated evaluation

---

Your program should produce an output log

A CSV-file with the tracking result

Format is defined on the project webpage

Ask your guide for details if needed

Competition: logs are parsed to generate a summary table for all project groups

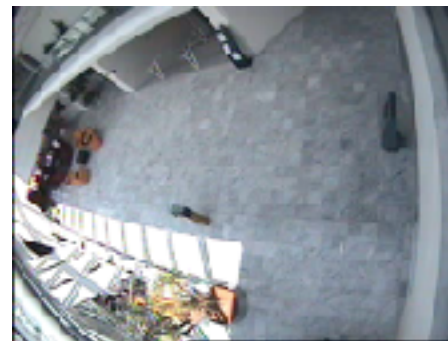
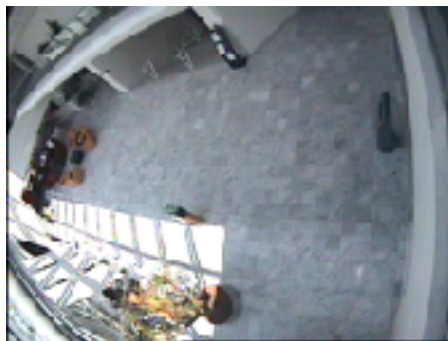
Results are to be produced for the input sequences.

If you downsample the input you need to compensate your output :-)

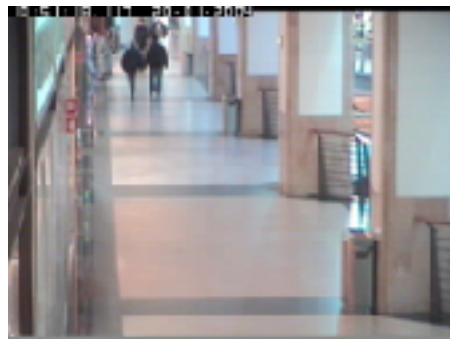


# Examples from CAVIAR

Camera 1: Wide field of view in a large room



Camera 2: Surveillance in a shopping mall





# Guidance

---

Each project group will have a guide

- a PhD student that gives guidance

- approx. 1h/per week

- meetings at times agreed upon by both parties

Guidance implies

- Give advice

- Give references to literature, code, and data



# GitLab @ IDA

---

More details: <https://www.ida.liu.se/gitlab>

NOT GitHub!

GitLab is administered using LIU-ID

Individual check-in, changes tracked with LIU-ID

Use the repository to store:

all external code used in the project

all code produced in the project

all documents produced in the project



# GitLab @ LiU

---

New option: LiU GitLab

Access point: <https://gitlab.liu.se>

Similar to Gitlab @ IDA

Has *continuous integration(CI)*:

automatic unit testing upon commit or push

Old option: Subversion(SVN)

ask project guide to have this set up



# Data storage

---

Each project group can get a separate file area

- 5 GB size

- No backup

- Deleted after project is finished

- If needed: ask your project guide



# Cooperation and plagiarism

---

## Discussions are allowed

Both between students and between project groups

## Declare code

Declare which code/text has been produced by the group and which is external

Put "own" and "external" code in different folders

Material declared as "own" must not be plagiarized from external sources

## Get external material approved

External material (e.g. code) must be approved

Allowed code packages are listed on web page

**Ask the examiner before using other packages**



# Design plan

---

Each group should produce a design plan:

- Describe the important functionalities (blocks)
- Describe data flow between the blocks
- Describe interfaces
- Describe methods of implementations
- To be approved by your guide

The plan should also assign responsibility:

- Each student must be responsible for the design of at least one major block
- Should be able to answer questions on that block during the final seminar





# Project management

---

Use reasonable level of **project management**

1. make a design plan together
2. discuss it with your guide
3. update to get it approved
4. divide work among project members

Each member must have an **assigned task**

1. Responsibility for a part of the technical work  
(but also cooperate)
2. Present that part at the seminar
3. Answer questions by the examiner and audience on that part



# Project context

---

## Produce test cases

Should be simple, easy to find expected result

Can be used automated testing at commit (at [gitlab.liu.se](https://gitlab.liu.se))

## Deliverables (examined individually and as a group)

A good presentation

A good report



# Test, test, test!

---

**Testing is one of the most important parts of the project**

Divide your system into parts that can be easily tested

You should test using *test data*

- Produces obvious or specified behavior
- Usually not the same data as the final system should operate on
- Can be synthetic data generated to produce well-known output from your system

Test all parts individually before integrating them

**Do not integrate parts into a larger system before they are properly tested**



# Deliverables

---

## A good presentation

Including examination of each project member

## A written project report

Which problem is solved?

How is it solved?

What is the result? Performance evaluation!

Why did you get this result?

References

Targeted to your fellow students, not doing this project  
(explain what you are doing!)



# Time table project 1, 2020

---

**Feb 7 (today):** formation of project groups

- Problem analysis

- Specification of required functionalities

- Selection of methods

- Who does what

- First contact with project guide

**Feb 14:** approval of design plan

**March 27:** hand in report, *approved* by guide

**March 31:** presentation of project results

Each student contributes ~80h (3hp) of work/project



# Implementation

---

The project can be implemented in:

- Python and/or C/C++ (hint: pybind11 for Python & C++)  
close to real-time should be possible (not a requirement)  
use pycharm IDE (installed in computer labs)
- Matlab  
real-time processing difficult

Several code packages exist:

e.g. OpenCV for C/C++/Python

See list on web page

External software for background modeling **not allowed**



# OpenCV

---

## Version 3.4.3

Installed in ISY computer rooms

## Contains many useful functions:

Image / video / camera input

Image display

KLT-tracker

Harris-operator

Morphology, statistics

## Requires C/C++/Python skills

See “OpenCV pitfalls” on course web page

Odd behaviour of some functions - always test external code!

Use Matlab otherwise (has similar stuff)



# Required functionalities

---

For all groups:

1. Tracking of objects in a simple sequence
2. Management of object identity
3. Evaluation of results using ground truth (includes logging)

Optional functionality:

1. Management of shadows and reflections
2. Management of spurious background motions (multi-modality)
3. Occlusion management  
e.g., prediction or ground plane modeling (camera hand over), foreground modelling, and KLT/optical flow

Groups of 5 students: 3 optional functions

Groups of 4 students: 2 optional functions