Practical Session: Person Tracking.

Ruben Martinez-Cantin, University of Zaragoza
rmcantin@unizar.es

Introduction

Object tracking in a video sequence is a relevant problem in the computer vision community, with a wide array of applications like surveillance, traffic monitoring, sports or video annotation. Object tracking is commonly modelled with dynamic graphical models, specifically with state-space models. In this practical we will implement an online solution for multi-person tracking in a video using the Kalman filter, analyze the results and comment on the most important elements of the design.

Objectives

1. Ground the theory on graphical models on a specific example of person tracking in a video sequence.
2. Understand the formulation of a tracking algorithm and its most relevant design parameters.

1 Problem Definition

Think about the problem of tracking a person in a video and formulate it using a graphical model; specifically a state-space model.

We will approximate the motion of the people by a constant velocity model in the image space plus a random acceleration noise, assuming that the camera is static and the person always upright. Let $(u \;\; v)^\top$ be the image coordinates of a person and $(\dot{u} \;\; \dot{v})^\top$ his velocity. The state-space model of the person at time t will be $x_t = (u_t \;\; v_t \;\; \dot{u}_t \;\; \dot{v}_t)^\top$. The transition or dynamic model g of the person will be given by the following set of linear equations

$$u_{t+1} = u_t + \dot{u}_t \Delta t + \epsilon_{\dot{u}} \Delta t \quad (1)$$
$$v_{t+1} = v_t + \dot{v}_t \Delta t + \epsilon_{\dot{v}} \Delta t \quad (2)$$
$$\dot{u}_{t+1} = \dot{u}_t + \epsilon_{\dot{u}} \quad (3)$$
$$\dot{v}_{t+1} = \dot{v}_t + \epsilon_{\dot{v}} \quad (4)$$

where $\epsilon_{\dot{u}}$ and $\epsilon_{\dot{v}}$ stand for the velocity impulse resulting from the acceleration noise, both normally distributed with zero mean $\epsilon_{\dot{x}} \sim \mathcal{N}(0, \sigma_{\dot{x}})$. Δt stands for the time between frames.

People can be detected in the image by using a HOG descriptor and an SVM classifier (we will use the standard approach of [2]). The measurement model h will be then given by

$$y_t^u = u_t + \delta_u \quad (5)$$
$$y_t^v = v_t + \delta_v \quad (6)$$

where δ_u and δ_v are the people detector noises in the image coordinates. Both noises are normally distributed with zero mean and a standard deviation of σ_u and σ_v.

- Draw the graphical model.
- Notice that the dynamic and measurement models are both linear. Write them in the form $x_{t+1} = Ax_t + B\epsilon$ and $y_t = Cx_t$.
- Write the Kalman Filter equations.
- Give a reasonable estimation of the acceleration and measurement noise standard deviations. Give a reasonable initial covariance for the state.

2 People Recognition

For this practical, we will use the dataset of [3, 1]. Download them from the web in the references. In any case, other videos can be used if they are suitable for the Matlab people detection algorithm that we will use. For the 1-person tracker it is advisable to use the dataset of [3]. Specifically, start with the sequence with identifier *2012-04-02_120351* in the collection named *Test_part01_bendingin.tar.gz*. You have such data in the lab drive (also in the link [1]).

We are going to use the popular OpenCV library for people detection. This library also includes functions for image handling (reading, writing, editing, etc.). Before the implementation of the person tracking algorithm, make yourself familiar with the image manipulation and people detection functions in OpenCV. Write a small program that runs the people detection algorithm in every image of the sequence. Some Python/OpenCV functions that can be useful are:

- `help(function_name)`: displays help text for the function `function_name`. If you use `ipython`, you can also use `function_name?`.
- `cv2.imread, cv2.imwrite`: reads/writes an image from the hard disk.
- `cv2.resize`: resizes an image object.
- `cv2.imshow`: shows an image on screen.
- `cv2.rectangle, cv2.drawMarker, cv.putText`: draws a rectangle, a marker or some text over the image.

For manipulating multiple files at the same time, you can use the `glob` standard library, which allows to use wildcards (for example: *.jpg) to generate a lists of paths.

For the people detection, we are going to use an SVM detector based on HOG features. The following functions and member functions can be used:

[1] `https://drive.google.com/file/d/1Mae0AaH8fCjta7Imti68vYaPGn_hAkOw/view?usp=sharing`

- `cv2.HOGDescriptor`: creates the detection object.
- `setSVMDetector`: detection object member function to select and configure the SVM detector. As a initial setup, you can use the default configuration from `cv2.HOGDescriptor_getDefaultPeopleDetector()`.
- `detectMultiScale`: detection object member function to detect the persons.

For optimal performance with the given datasets, it is recomended to reduce the images size (while keeping the original aspect ratio) and tweak the parameters `winStride, padding, scale` of the multi-scale detector. You can also see the associated score with each detection and filter some bad detections.

For the report, it might be useful to show some plots and graphs. For that, you can use `matplotlib.pyplot` which behaves very similar to Matlab plotting.

3 1-Person Tracker

Now let's implement a 1-person tracker. In the first frame apply the person detector algorithm and select one of the detected persons as the person to track. For example, take the first one detected.

In the rest of the frames, apply the Kalman filter equations

- *Prediction*. Use the prediction equations to predict the person position in the image at time $t+1$ based on the position at time t.
- *Matching*. Find the person detection that corresponds to the person that is being tracked. Matching is a key aspect in tracking. Notice that, if you run the people detector in the image I_{t+1}, in some frames there will be several persons detected. Which one is the one that we are tracking?

  There are several approaches that can be used here. As it is not the focus of this practical, we will follow a nearest-neighbour strategy. Observe that, from the prediction step, you have a prediction for the position and the uncertainty of the person. Take as the correspondence the detected person that is closest to the prediction, only if it falls inside the uncertainty predicted by the Kalman filter.
- *Update*. Use the Kalman update equations to fuse the prediction and person detection informations.

Display the results. Very importantly; display the prediction, detection and update person positions in different colors (e.g., red for prediction, green for detection and blue for the update). Observe the effect of changing the noise covariances. Understand the meaning of the covariances and the effect of its relative sizes.

Show also in the image the covariance ellipses for the position, using the function `plot_ellipse` given as extra material for the practical. Compare the prediction and update covariances.

Notice that you might want to change the covariance noises of your Kalman filter based on what you observed in your experiments. This is frequently known as tuning.

Observe that, when the person goes out of the image, the tracking should terminate. Implement a reasonable tracking deletion policy.

Draw the trajectories of the predicted, updated and detected persons. Observe how the noisy person detections are filtered by the Kalman filter.

4 Multi-Person Tracker (Optional)

Now that your 1-person tracker is implemented and tuned, let's implement a multi-person tracker.

For that, you will have to assign each person an identifier and maintain a Kalman filter for each one of them separately. First of all, assign each person an identifier apart from the Kalman filter parameters. You can use the function `cv2.putText` to display the person identifier next to the person in the image.

After that, all you have to implement are the policies for person appearance and disappearance. A person appears and has to be tracked when a person is detected and it is not the correspondence of any existing tracked person. A person disappears when it is not detected in the image for a number of frames.

5 Report

The report of the practical should be uploaded to Moodle. The focus of the report should be the theoretical aspects of the practical and not the implementation details. The results should be presented as formally and graphically as possible. Upload the report in a zip file and attach the Python files that you used in the practical and any other file that you consider relevant.

References

[1] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008. http://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/people-detection-pose-estimation-and-tracking/people-tracking-by-detection-and-people-detection-by-tracking/.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[3] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *Pattern Recognition*, pages 174–183. Springer, 2013. http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Pedestrian_Path_Predict_GCPR_1/pedestri