DOAS 2007 project
Looking-at-people: intelligent surveillance systems

Wojtek Zajdel
wzajdel@science.uva.nl

1 Introduction

Intelligent Autonomous Systems increasingly often operate in environments inhabited by humans, like houses, public places (stations, shopping centers) or intelligent vehicles. In order to properly interact with people an intelligent system has to detect people in its environment, identify them and possibly recognize their behavior. Examples of such systems are intelligent security systems that detect people entering restricted areas or detect aggressive behavior; intelligent vehicles that detect pedestrians (and warn the driver if a pedestrian is to close to the vehicle), or interface robots that acts as an interface for taking user's commands and presenting results.

Perception of humans and their actions typically relies on cameras as sensors which capture various details about the location, number and appearance of people. Therefore computer vision techniques for "looking-at-people" are the mainstream of research in Intelligent Autonomous Systems. The relevant algorithms range from low-level image processing techniques (edge detection, pixel classification) to higher-level gesture and object classification methods.

The DOAS 2007 project will be performed in the context of larger research at the ISLA laboratory which develops computer vision techniques for "looking-at-people". Currently, the laboratory runs several projects that aim at detecting people in still images, tracking people through images sequences, detecting at tracking individual body parts or pose estimation of humans in 2D and 3D. In our project of interest, CASSANDRA [4], the final goal is to automatically detect aggressive behavior of people in public spaces like train stations.

The laboratory provides an opportunity for students to get hands on experience with several advanced image processing tools. Additionally, there are several realistic video sequences involving professional actors recorded in real-world setup (train station). These dataset will be used in the project to evaluate the developed algorithms.

2 Project description

The project will study computer vision and pattern recognition techniques for detecting humans in still images. Specifically, we will follow a "bottom-up" approach for detection that can be seen as a two stage algorithm. In the first stage, a series of detectors is deployed in order to detect loose human body parts, easily distinguished from other objects in an image. Examples of such body parts are faces, head-shoulder contours, full upper-body contours (torso-head-arms). In the second stage, an assembly algorithm groups the detected parts into an full-body configuration.

2.1 Task

The idea is to first use a series of part detectors (already available) to find loose human body parts in an image. Next, the loose body parts need to be assembled into a valid configuration that represents a person. At this moment there are four detectors available:

1. face detector (based on [9], implementation from OpenCV)

Figure 1: Examples of part detector output: (left) faces (right) full-body contours.

2. head-shoulder and limb detector (based on [5], IAS implementation)
3. upper-body detector (based on [9], implementation from OpenCV)
4. full-body detector (based on [9], implementation from OpenCV).

Each of the detectors indicates image regions where a hypothetical body part is located (see Figure 2.1).

Therefore, the primary objective is to develop an algorithm that takes the responses of individual detectors as input, groups these responses into a valid human configuration and outputs image regions where a person is located. The key challenges for the grouping problem are

- inaccuracies of individual detectors: sometimes a genuine body part will not be detected, and sometimes a false body part will be indicated by a detector.
- occlusions: sometimes certain body parts are not visible, due to occlusion by other objects in the scene or by other body parts
- dealing with multiple people: when several people appear close to each other in an image, there will be multiple responses from each detector. The challenge of the assembly algorithm is to properly match different body parts to different people.

An additional task in the project is to develop a simple labeling tool that allows an user to mark image regions (bounding boxes) where a person is visible. Such human annotated bounding boxes will be used as a ground-truth for evaluating the performance of individual detectors (existing) and the aggregate algorithm (to be developed).

2.2 Plan

1. Read relevant material for Chamfer system [5], hierarchical object detection [9], assembly-techniques [1, 3, 8], and survey material [6].
2. Download and install OpenCV library [2]. Get familiar with the programming with the library [10].
3. Practice usage of part detectors. Run simple tests on several images. Make a module that can be later integrated with the aggregated algorithm
4. Develop (very simple) labeling application. The application should load a series of images (e.g. all *.ppm files from a given folder), present a GUI that allows a user to draw rectangles, and save parameters of the rectangles. For each image: number and coordinates of all rectangles in a text of Matlab file.
5. Label around 200 images (2 minutes/image x 200 images = 7 hours = 3 days x 2.5 hours/day). It is recommended that this step is performed quite early in the project, since the evaluation of detectors requires labeled images.
6. Given labeled images, test individual detectors in a systematic setup (ROC curves)
7. Design/implement/test assembly algorithm (Several versions are welcome).
   - Start simple. Use the evaluation results of part detectors to decide which parts are most reliable.
   - Think of constraints on distance and size of one part relative to the other.
   - Think of color (appearance) similarities between body parts.
   - See [1, 8, 7] for inspiration.
8. Perform systematic tests on labeled images. Compare results with individual detectors. Compare processing time.
9. Write paper, prepare a presentation.

2.3 Tests and Implementation

There will be video sequences provided. The sequences have been recorded at a train station and present several (1-5) people in each frame. The poses of people range from normal standing, walking, to quite unnatural like jumping, lying, kicking.

The primary evaluation criterion for detection algorithm will be ROC curves that measure detection accuracy and false-positive rate against ground truth. A particularly important will be comparison of the aggregate detection algorithm vs part detector.

All algorithm should be implemented in MS Visual Studio C++ environment. The individual body part detectors have been tested in this environment.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.

[2] Intel Corporation. Open Source Computer Vision Library, 2006. `http://www.intel.com/technology/computing/opencv/index.htm`.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[4] D. Gavrila. http://www.science.uva.nl/research/isla/themes/perception/cassandra/.

[5] D. M. Gavrila and V. Philomin. Real-time object detection for smart vehicles. In *IEEE International Conference on Computer Vision*, pages 87–93, 1999.

[6] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[7] A.S. Micilotta, E.J. Ong, and R. Bowden. Tracking of humans by probabilistic body part assembly. In *Brtish Machine Vision Conference*, 2005.

[8] G. Mori, X. Ren, A.A Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Computer Vision and Pattern Recognition*, pages 326–333, 2004.

[9] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 7(2):137–154, 2004.

[10] G. Wilson. Programmer's Toolchest. The OpenCV Library Dr.Dobbs Journal, 2001. http://www.ddj.com/dept/architect/184404319.