# AE3B33KUI — (feature-based) classification seminar

authors:
Tomáš Svoboda, Tomáš Werner, David Hurych et al.[*]
http://cmp.felk.cvut.cz, 2008

seminar teachers for SS 2012/2013:
Pavel Vostatek[†], Eduard Bakštein, Tibor Strašrybka,
Radek Píbil, Radek Mařík, Matěj Holec,
Martin Selecký, Jiří Anýž

February 18, 2013

This is a supplemental material for the (feature-based) classification and decision making under uncertainty for the bachelor's track X33KUI seminars.

Assignments for either first or second seminar are just suggested, not mandatory.

For those interested in a more detailed understanding, we suggest books [1, 3] that are easily readable even for a bachelor student. Book [5] is slightly more complicated, but supplies an excellent, deep knowledge on the subject. For practical task, we suggest the Matlab toolbox [4], which can be acquired for free.

## Contents

[*]This learning material is partially based on an older material by V. Franc, T. Pajdla and T. Svoboda. It has also been inspired by some assignments from the X33RPZ course. A part of the data has been generously provided by the Eyedea Recognition company. The authors would also like to thank Jiří Matas for productive discussions.

[†]If you have any questions, please contact your teacher first. If you are not sure who is your teacher then email to vostapav@fel.cvut.cz

# 1 Seminar 1

## 1.1 Bayesian decision-making

**Exercise 1.1** We know the statistical distribution of male and female heights (see the following table). Infer the most likely gender of a person 168 cm (L) tall.

| $x$ | XS | S | M | L | XL | XXL |
|---|---|---|---|---|---|---|
| | (0–100) | (100–125) | (125–150) | (150–175) | (175–200) | (200–$\infty$) |
| $P(x\|\text{male})$ | 0,05 | 0,15 | 0,2 | 0,25 | 0,3 | 0,05 |
| $P(x\|\text{female})$ | 0,05 | 0,1 | 0,3 | 0,3 | 0,25 | 0 |

**Exercise 1.2** Solve this task for a person that has been picked at random from a group with 60% of males and 40% of females.

**Exercise 1.3** How would you find the probabilities of the conditional distribution $p(x|s)$. What about the apriori case – $p(s)$?

**Exercise 1.4** Can we possibly find an error-free decision strategy (make no mistakes)? For what group of people is the inference based on height of a person particularly bad?

**Exercise 1.5** Propose a classifier for a gender inference, based on two suitable measurements of a person. Discuss the quality of the possible solutions.

**Exercise 1.6** We have a bag of old coins that have been subject to various levels of use. Coins of different values may therefore be of different sizes. The value of a coin can be still observed by looking at it. We have a task to partition the coins to the groups of same value according to their values based on their weight. We know that there are coins of values $1, 2, 5$ Kč, that is $s \in 1, 2, 5$. The loss function is:

$$l(s, d) = \mathrm{abs}(h_d - h_s) ,$$

where $h_s$ is the value of a selected coin and $h_d$ is our decision about the value of the coin.

We have fast and simple to use scales with a 5g precision. Suggest a classification strategy that would minimize the loss and work. By work we mean actually classifying the coins manually. This may be the case if we have to achieve an absolute precision. This is not possible for a vending machine though, as it may accept coins of various values. Discuss the effect of manual labour on the cumulative loss caused by bad classifications.

Let's estimate the expected weight of each of the coins based on an experiment. Pick 100 coins, weigh them and record their values. As a result, we are going to have a training multiset. We may end up with a following table:

| $s \mid x$ | 5 | 10 | 15 | 20 | 25 | $\sum$ |
|---|---|---|---|---|---|---|
| 1 | 15 | 10 | 3 | 0 | 0 | 28 |
| 2 | 7 | 13 | 16 | 6 | 1 | 43 |
| 5 | 0 | 1 | 2 | 11 | 15 | 29 |

How many possible strategies are there? Our scales say the coin is 10g, which class would you choose?

**Exercise 1.7** You are given a $10 \times 10$ pixel image, and each of the pixels can have one of 256 possible values for intensity. How many features are provided? How many possible values are there for the whole set of features (values in the observation space)? How many possible strategies are there?

You are given an image, and you are told that there is a letter in it $S = \{\mathrm{A}, \ldots, \mathrm{Z}\}$. How would you represent and estimate $P(\vec{x}|s)$?

**Exercise 1.8** Propose a classifier for the previous exercise, which assumes features to be conditionally independent.

## 2 Seminar 2

**Exercise 2.1** Prove that there exists a strategy minimizing the expected risk (loss), which identical to a strategy maximizing $P(s|\vec{x})$ for the following loss function:

$$l(s, d) = \begin{cases} 0 & \text{iff } d = s \\ 1 & \text{iff } d \neq s \end{cases} . \tag{1}$$

### 2.1 Nearest neighbour classifier

**Exercise 2.2** Create a single nearest neighbour classifier (1-NN) for a person's gender given with *height* and *age* as features. If we changed units for *height*, would the classifier change?

**Note:** 1-NN is a relatively good classifier, even though it is quite simple. If we know the actual p. d. [2], then it is possible to show that the probability of a wrong classification is at most twice as bad asymptotically as the size of the training multiset approaches $\infty$.

## 2.2 Linear Classifier

Another way to avoid the need to estimate and represent probabilities is to design discrimination functions $g_s(x)$[1] directly. One of the options is the linear classifier, which uses discrimination function in the following form:

$$g_s(\vec{x}) = \vec{w}_s^\top \vec{x} + b_s \,, \tag{2}$$

where $\vec{w}_s$ is the weight vector and the scalar $b_s$ for a translation[2] along the y-axis (also called a bias). An object $\vec{x}$ is classified as a member of the class $s$, whose discrimination function $g_s(\vec{x})$ values is greater than those of the other classes. Therefore, this task transforms into an optimization problem, where we are looking for parameters of the linear discrimination functions, which minimize some criterion, e.g., number of wrong classifications on the given training multiset.

**Exercise 2.3** Prove that a set of feature vectors a linear classifier classifies into a single class is convex.

**Exercise 2.4** Is it possible for the training multiset in the Figure 1 to be classified by a linear classifier perfectly? If so, sketch such a linear classifier.

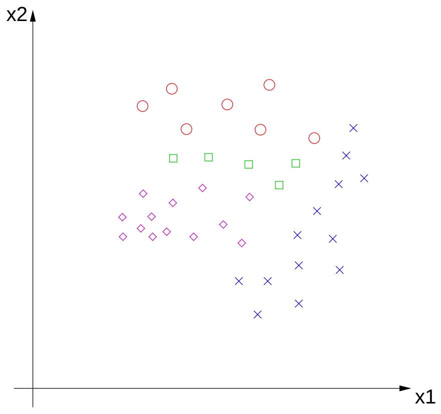

Figure 1: Training multiset for a 2D feature space with 4 classes.

[1]Discrimination functions exists even for Bayesian classifiers. Discuss.

[2]We have used slightly different naming conventions in the lectures $g_s(\vec{x}) = \vec{b}_s^\top \vec{x} + c_s$.

## 3 Programming exercise – alphanumeric characters recognition

### 3.1 Task Specification

The task is to classify some letters of car registration plates. Let's assume that a plate has already been localized in the picture (see Figures 5 and 6). Letter images are normalized to be $10 \times 10$ pixels. You have a randomly chosen training data at your disposal. Letter images are represented by row vectors of brightness values of concatenated columns of pixels (see Figure 3). Each row in the input text files represents one image. The `train.txt` file contains feature vectors and the `train_labels.txt` file the respective class labels. Images are just for convenience, the important data is in the ASCII text files. The result of classification should be a text file with classifications. Each row should contain a class for the letter on the same line in `train.txt`.

It is important to realize that this is usually everything what a customer has available. After you claim you are ready with your code, your customer, seminar teacher in this case, uses testing data to evaluate your work. The testing data are going to be located in files `test.txt` and `test_labels.txt`. We recommend that you simulate this situation by splitting the data you were provided into your own training and testing multiset.

### 3.2 *Individual Exercise*

Implement the following algorithms in order to complete the exercise explained in the previous paragraph.

**A Nearest Neighbour Classifier** Implement a 1-NN classifier. Show that it performs this task on the training multiset.

**Bayesian Classification** Implement a naïve Bayes classifier. Assume conditional independence of intensities for each of the pixels. Therefore, we have the following formula for each of the classes:

$$P(\vec{x}|s) = P(x_1|s)P(x_2|s)\dots P(x_n|s) = \prod_{i=1}^{n} P(x_i|s) \,, \tag{3}$$

where $x_i$ represents the intensity of the $i-$th pixels.

### 3.3 Example Solution – The Perceptron Algorithm

You have one example solution at your disposal, which uses a perceptron for classification. A usage example can be found in the `main` method of `MainFrame.java`.

The general class `Classifier` contains the `load_matrix` I/O method for the feature vectors loading, `load_labels` I/O method for the correct classification and `count_confusion` classification evaluation. Each of the classifiers is a special class, which inherits the classifier class. The basic methods are `learn` and `classify`.

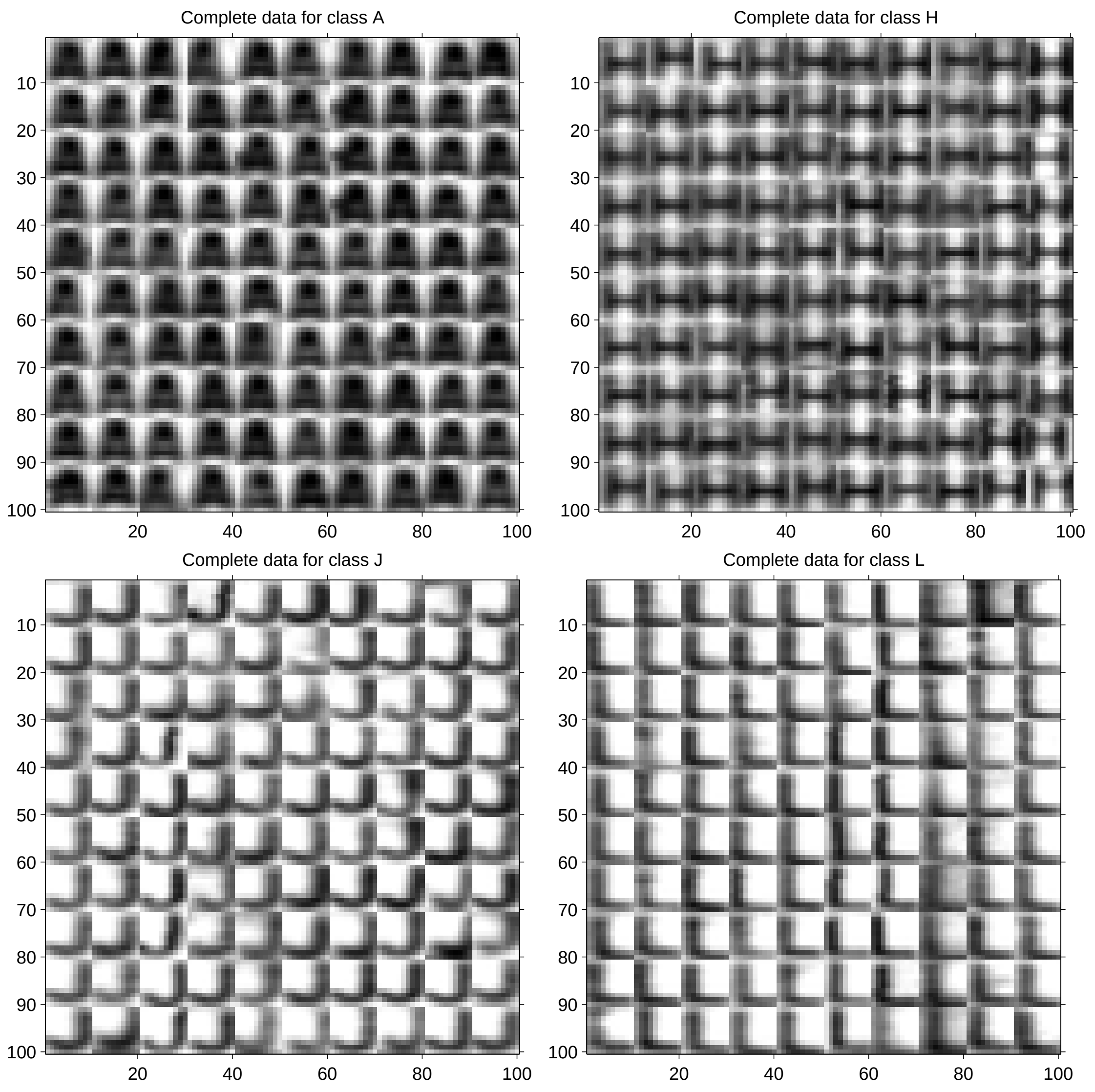
Complete data for class A
Complete data for class H
Complete data for class J
Complete data for class L

Figure 2: Examples of normalized letter images of registration plates.

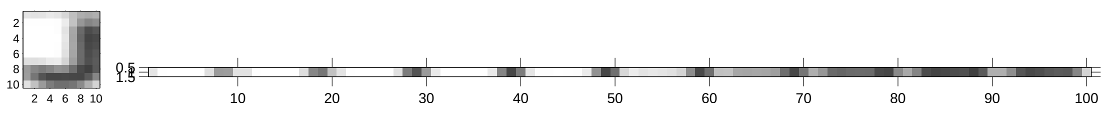

Figure 3: Pixels are represented by a row vector of concatenated columns. First comes the first column, then the second etc. It is obvious that dark columns of the letter J, which are in the extreme right part of the image, are at the end of the data vector.

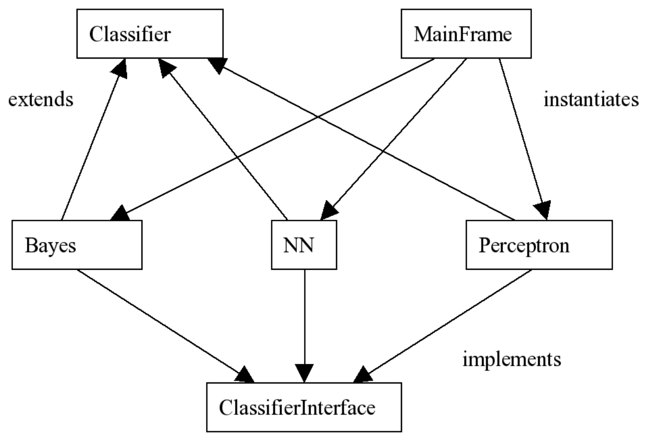

Figure 4: Class diagram.

## 4 Auxiliary code description

### 4.1 `Classifier` Class

This class contains data structures common to all classifiers. Each of the concrete classes of classifiers (Bayes, nearest neighbor, perceptron) inherit it.

#### Methods of the `Classifier` Class

`load_matrix(String fileName)` – method implements the training or testing data loading from a file with a name passed as the parameter. The data is then stored in the `matrix` field.

`load_labels(String fileName)` – method implements the training or testing data loading from a file with a name passed as the parameter.

`count_confusion()` – this method checks first whether is the classification finished already. Then this methods checks vector lengths of the suggested labels and then it compares these vectors against the contents of the file loaded by the `load_labels()` method.

### 4.2 `ClassifierInterface` Interface

This interface defines the `learn()` and `classify()` methods. Every classifier must implement this interface.

## 4.3 `MainFrame` Class

`MainFrame` is the main class, which contains the `main()` method, which allows for the instantiation of the concrete classifiers and calling their learning and classification methods. This class also contains methods for serialization/deserialization of the object-classifiers and allows for writing/reading of these objects along with their learnt parameters. Therefore, it is not necessary to perform time consuming classifier learning every time, there is a classification to be performed.

### `MainFrame` class methods

`main()`

`saveClassifier(String fileName, Classifier c)` – this method implements the serialization (writing) of an instance of a concrete classifier `c` into the file name `filename`.

`loadClassifier(String fileName)` – this method implements the deserialization (reading) of an instance of a concrete classifier `c` from the file name `filename`.

## 4.4 `Perceptron` Class

This is the concrete implementation of the Perceptron classifier. This class inherits `Classifier` and implements the interface `ClassifierInterface`.

### `Perceptron` Class Methods

`learn()` - this methods verifies whether the data has been loaded into the `matrix` and `labels` variables, so that it has something to work with. There is also a preparation phase, which counts the number of possible classifications. Classifications are mapped to number and the matrix `w` and vector `b` that represent the classifier parameters. Finally there is the learning itself as per the following algorithm:

1. Set $\vec{w}_y = \vec{0}$ and $b_y = 0$ and for all $y \in Y$ ($Y$ – set of all possible labels).
2. Pick a random incorrectly classified input. If there is no such input, then STOP, because the learning has finished, by finding parameters for an error-free classification of the input data.
3. 3. Let $(\vec{x}_t, y_t)$ is an improperly classified input and $\hat{y}$ is a classification of $\vec{x}_t$ using the current classifier. Adapt the parameters of the classifier according to the following formulae:

$$\begin{aligned} \vec{w}_{y_t} &= \vec{w}_{y_t} + \vec{x}_t \\ b_{y_t} &= b_{y_t} + 1 \\ \vec{w}_{\hat{y}} &= \vec{w}_{\hat{y}} - \vec{x}_t \\ b_{\hat{y}} &= b_{\hat{y}} - 1. \end{aligned}$$

4. Continue with step 2.

`classify()` – this method checks whether the classifier has finished learning and whether data structures are valid. If no problem has been found, classification follows according to

$$\hat{y} = \arg\max_{y \in Y} \vec{x}_t^\top \vec{w}_y + b_y$$

with the result being recorded in `result_labels` and sent to the standard output.

### 4.5 `NN` Class

A class for the nearest neighbour classifier ready for implementation. The result of the classification must then be saved into the `result_labels` vector (so the confusion table can be calculated).

### 4.6 `Bayes` Class

A Bayes classifier class that is ready for implementation. The result of the classification must then be saved into the `result_labels` vector (so the confusion table can be calculated).

## 5 Evaluation

- Implementation of the 1-NN classifier [0–3] pts.
- Implementation of the Bayes classifier given the conditional independence of the probabilities [0–6] pts.
- Robust implementation capable of handling arbitrary number of classes or feature vectors [0–1] pt.
- Bonus points for an effective, interesting implementation and additional features [0–1] pt.

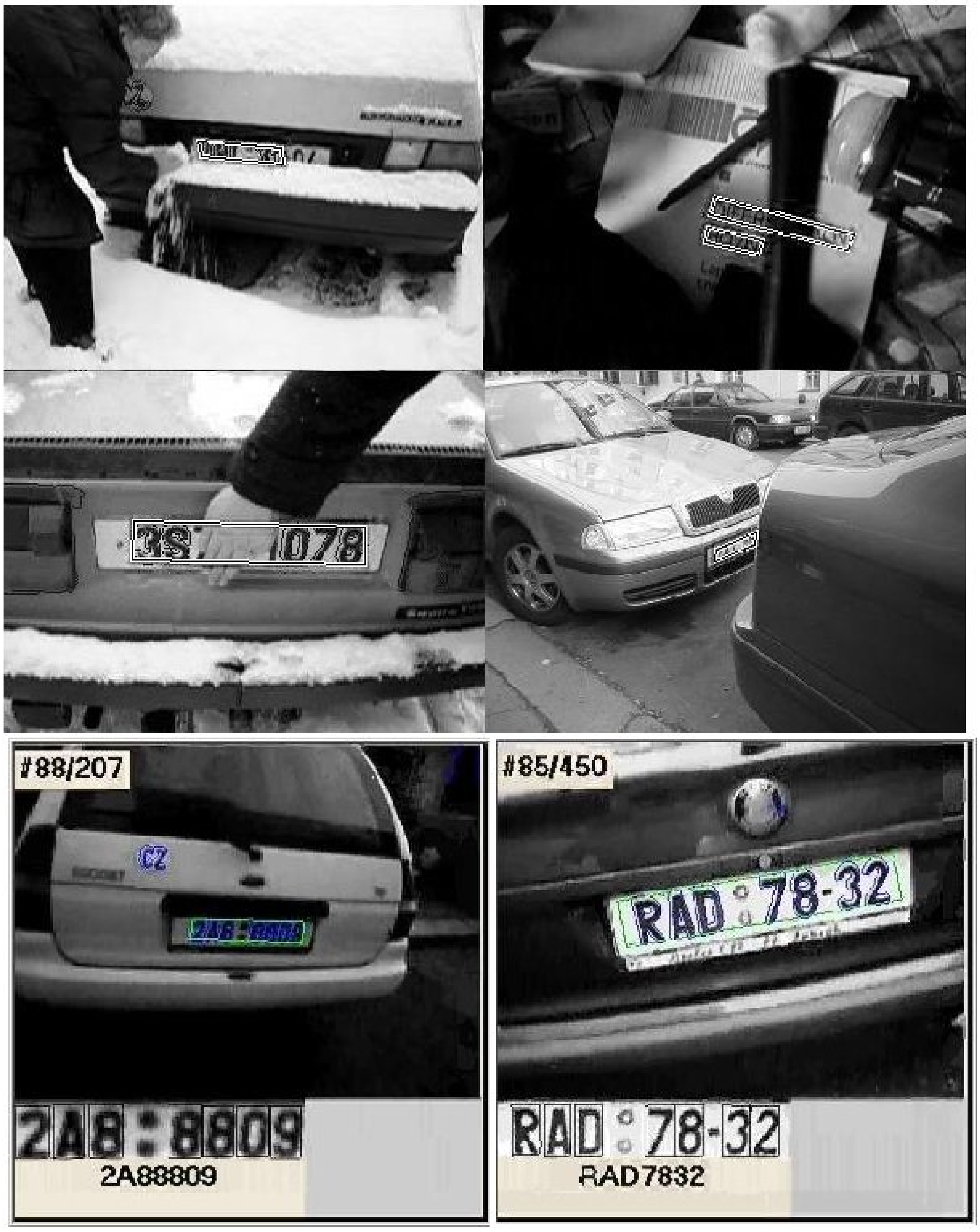

Figure 5: Example of an automatic text localization in an image. Further information can be found at `http://cmp.felk.cvut.cz/~zimmerk/lpd/index.html`

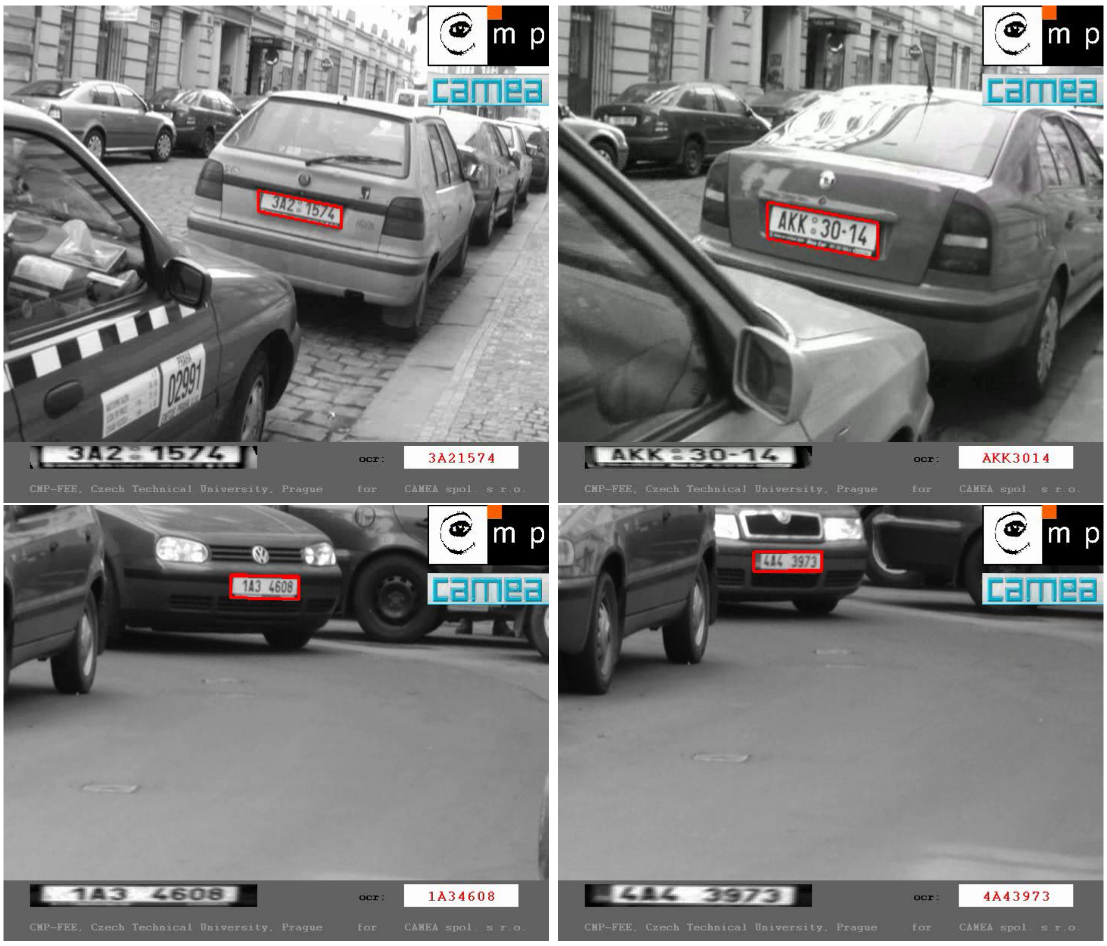

Figure 6: Example of a commercial application for registration plate recognition in a video. Demo videos can found at http://cmp.felk.cvut.cz/cmp/courses/X33KUI/Videos/RP_recognition/ and you have also seen a demonstration of the first algorithm in the first seminar.

## References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Bussiness Media, New York, NY, 2006.

[2] T.M. Cover and P.E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification.* Wiley Interscience Publication. John Wiley, New York, 2nd edition, 2001.

[4] Vojtěch Franc and Václav Hlaváč. Statistical pattern recognition toolbox for Matlab. Research Report CTU–CMP–2004–08, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, June 2004. `http://cmp.felk.cvut.cz/cmp/software/stprtool/index.html`.

[5] Michail I. Schlesinger and Václav Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.