People detection and tracking on video

Autor:

Mateusz Szumilas

Supervisor: Ing. Vitěslav Beran, Ph.D.

Brno, May 2019

Contents

Abstract

This work will show and describe the algorithms used for detection and tracking objects on films without the use of neural networks. The work presents algorithms such as AdaBoost, an algorithm that creates one strong and many haul-cascades, to detect through predefined patterns. At work, we do not experience algorithms in which neural networks exist. In this work I would like to show algorithms that are less popular, however, it is worth looking at them for the sake of history. Currently, other, much faster and more accurate algorithms are often used, but it is worth looking at it from the perspective of time and seeing how earlier such have been dealt with.

Chapter 1

Haar-cascade

1.1 Detecting by Haar-cascade

1.1.1 Description

To detect by this method we used a set of classifiers. Which are they looking for so-called Haar's features. These features contain information about changing the contrast value between rectangular groups. These features can be easily scaled and rotatable. To speed up the calculation, a matrix is created from pixels, where each field is the sum of all values from the left and over the pixel. As a result, we always do only 4 operations while checking the classifier. What is the complexity of O(1).
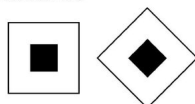
Edge Features

Line Features
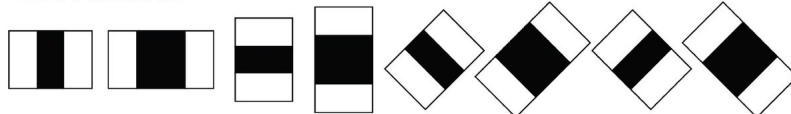
Centre-surround features

1.1.2 Description of the action

This classifier distributes pixels to black and white rectangles. It then checks if the area is an edge, a line or a central point. Next, the substraction of the sums of bright and dark pixels is calculated and determines whether the given field matches the pattern.

$$\Delta = dark - white = \frac{1}{n} \sum_{dark}^{n} I(x) - \frac{1}{n} \sum_{white}^{n} I(x)$$

Where:

- Dark means all pixels in black part of classifier.

- White means all pixels in white part of classifier.

- n means number of pixels

- I(x) means value of this pixel

In perfect case Δ will be equal 1

Comment 1.1.1. In Haar-cascade we are using a hundreds and even thousands of classifiers. Most often combined into one using AdaBoost.

Chapter 2

AdaBoost

2.1 AdaBoost

AdaBoost is an algorithm that has to do from many weak classifiers, create one arbitrarily strong classifier. The algorithm in k-th iteration checks k weak classifiers on a set of examples D with variable weights

Weak Classifier 1   Weak Classifier 2   Strong Classifier
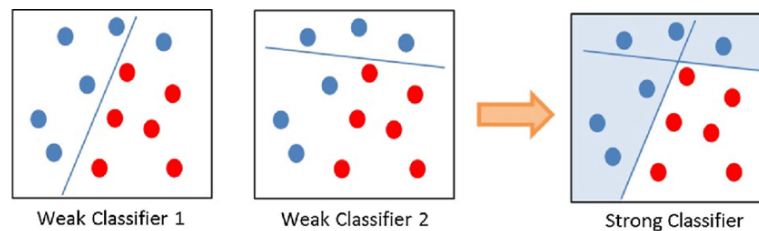
Definition 2.1.1. " The "weaker" classifier is a relatively simple classifier, with a very low expression power, able to classify test data with an efficiency greater than 50% "[4]

2.1.1 Algorithm

1. The input algorithm receives the training set (x_1, y_1),(x_2, y_2),...,(x_n, y_n), where each x_i should perform the appropriate X problem (e.g. detecting a car), and each label (decision) y_i should be made a certain set of Y. Let us assume that every y_n from $Y \in \pm 1$. Let us mark our set of classifiers K $(f_1, \ldots, f_K)$. to our weaker classifications.

2. At the beginning of the algorithm's operation, all weights in this algorithm are equal. Let $(d_1, \ldots, d_n)$ will be the set of weights for $(x_1, \ldots, x_n)$, respectively. At the beginning $d_i = \frac{1}{n}$, where i = 1, ...,n. We also remember the condition that:

$$\sum_{j=1}^{n} d_j^{(m)} = 1$$

3. AdaBoost calls the selected "weak" learning algorithm. We assume that the error of the classifiers obtained on the training set is less than $\frac{1}{2}$. So we teach the given algorithm f_m on X.

4. Then we calculate the received weighted error of the learning algorithm. We use the following formula for this:

$$err_m = \sum_{j=1}^{n} d_j^{(m)} [f_m(x_j) == y_j]$$

The task of the "weak" learning algorithm is to build a hypothesis f_m: X→Y appropriate for the current distribution D_t.

5. Then we calculate the weight of the classifier. It is interesting that if the classifier wrongly classifies an example, the badly classified by its predecessors then it will get a worse rating

$$w_m = \frac{1}{2} \cdot \log(\frac{1 - err_m}{err_m})$$

6. When AdaBoost gets the classifier f_m, the w_k parameter is selected. Intuitively, w_k is responsible for the importance we attach to the classifier f_m. Note that $w_k \geq 0$ when $err_k \leq \frac{1}{2}$. In addition, w_k grows when err_k decreases.

$$d_j^{(k)} = \begin{cases} d_j^{(k)} e^{-w_k}, if\, f_k(x_j) = y_j \\ d_j^{(k)} e^{-w_k}, if\, f_k(x_j) \neq y_j \end{cases}$$

7. The distribution of D_t is then changed to increase (decrease) the weight of the training set elements that are bad (well) classified by f_m. Hence, weights tend

to focus on "difficult" examples. We must also remember about the first condi-
tion. So we have to normalize the result:

$$s_k = \sum_{j=1}^{n} d_j^{(m)}$$

$$d_j^{(k+1)} = \frac{d_j^{(k)}}{s_k} \quad j = (1, \ldots, n)$$

2.2   Lucas-Kanade method

2.2.1   Optical flow

Optical flow is a set of vectors describing the movement of an object on subsequent
images. We distinguish three main methods of finding optical flow: - gradient
method - frequency method - correlation method The biggest problem with opti-
cal flow is lighting, which very much affects the results. One of the most popular
algorithms is the Lucas-Kanade algorithm.

2.2.2   Method LK or optical flow

It is a method of optical flow, which means that it looks for a pattern of apparent mo-
tion. This method involves examining the gradient between consecutive images. He
also assumes that the difference between the images is small and that the intensity of
the pixels does not change. Unfortunately, but the pattern has more unknowns than
equations. What makes it impossible for us to solve it, fortunately is the way to do
it.

$$I(x, y, t) = I(x + \delta_x, y + \delta_y, t + \delta_t)$$

After transforming this formula from the right with the use of Taylor's theorem and
after its over-conversion, we get the following pattern:

$$\frac{\Delta I}{\Delta x} v_x + \frac{\Delta I}{\Delta y} v_y + \frac{\Delta I}{\Delta t} = 0$$

Where I'_x, I'_y, I'_t are the gradient of rank in each direct, and v_x and v_y are values of moving in a given direction. Unfortunately, we still have two unknowns, which makes it impossible for us to calculate an unambiguous value. However, if we add the additional, that the neighborhood of each pixel does not change, it allows us to store subsequent points, then using the least-squares method, we are able to obtain the following formula:

$$Av = b \quad where:$$

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \cdots & \cdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \quad b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \cdots \\ -I_x(p_1) \end{bmatrix}$$

Using the method of least squares we obtain the following formula:
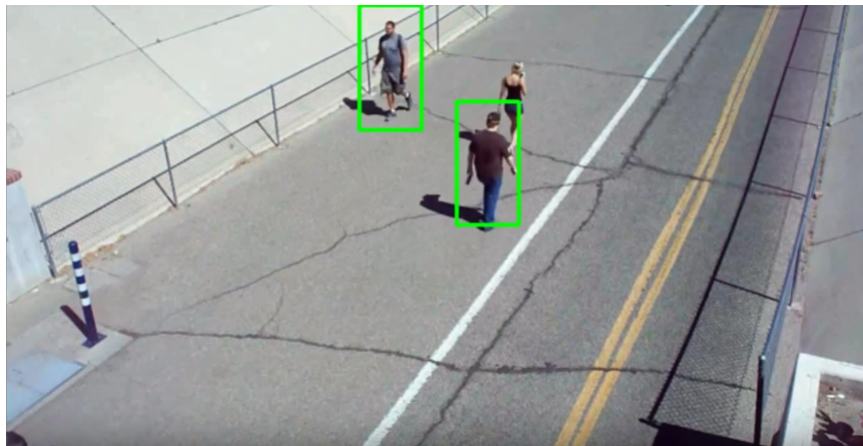
$$V = (A^T A)^{-1} A^T b$$

Chapter 3

Implemetation

3.1   Iplementation

To solve the problem I used the Python programming language, and the OpenCV library which greatly facilitated the work on the project.
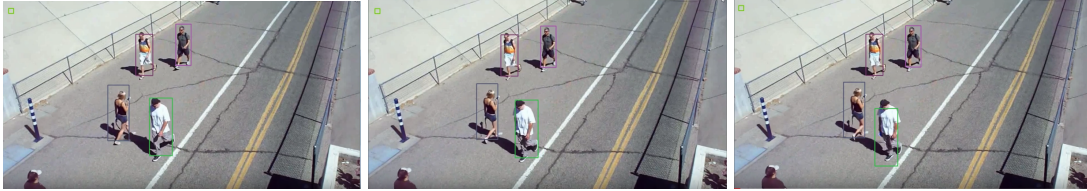
3.1.1   Detection

To detect people on film, I used the built-in detectMultiScale() function, which operates based on Haar-cascade and Adaboost

3.1.2 Tracking

For the tracking of objects, I also use functions built into openCV which is Tracker-Boosting_create() based on similar rules as the previous function. This function can have big problems when tracking many objects.

Summary

The solutions used are not very effective and slow, but I think that it is worth to know them because of the influence on stories and interesting solutions.

Bibliography

[1] Adam Dąbrowski Damian Jackowski, Tomasz Marciniak. *IMPLEMENTACJA I ANALIZA MODELI DETEKCJI I ŚLEDZENIA OBIEKTÓW W SEKWENCJACH WIDEO Z ZASTOSOWANIEM MODUŁU Z PROCESOREM SYGNAŁOWYM*. Foundation of Computer Science, January 2010.

[2] dr inż. Michał Grochowski. Rozpoznawanie wzorców i twarzy.

[3] Łukasz Kucharczyk. Wykrywanie i śledzenie twarzy na obrazie z kamery, 2011.

[4] Damian Łoziński. Zespoły klasyfikatorów, 2008.

[5] Miron Moderau. Sterowanie animacją przy użyciu kamery internetowej, 2011/2012.

[6] Dhara Patel. *Optical Flow Measurement using Lucas kanade Method*. Foundation of Computer Science, January 2013.

[7] Krzysztof Sopyla. Przetwarzanie obrazu z wykorzystaniem splotu funkcji.

[8] Piotr Wilkowski. Wykorzystanie algorytmu detekcji i lokalizacji w zadaniu chwytania, 2009.