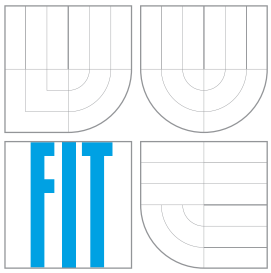
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANALÝZA POHYBU OSOB STACIONÁRNÍ KAMEROU

ANALYSIS OF MOTION OF PEOPLE BY A STATIONARY CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

STANISLAV SMATANA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ADAM HEROUT, Ph.D.

BRNO 2014

Abstrakt

Hlavním cílem této práce bylo navrhnout a vytvořit systém sledování osob s aplikací v oboru bezpečnosti nebo pro analýzu chování zákazníka v obchodě. Systém byl úspěně implementován pomocí metod KLT trekování, AdaBoost klasifikátoru a datové asociace pomocí Markovských řetězců a metody Monte Carlo. Implementace umoňuje analýzu pohybu lidí ve vnitřních i vnějích prostorech.

Abstract

The main goal of this thesis is to develop multi-target tracking system for use in field of security surveillance or for customer behavior analysis. The system was successfully implemented using KLT tracking, AdaBoost classifier and Markov Chain Monte Carlo data association. It is able to perform analysis of motion of people in both outdoor and indoor environment.

Klíčová slova

sledování více cílú,sledování pomocí detekcí, monitoring, MCMC datová asociace, analýza pohybu

Keywords

multi-target tracking, surveillance, tracking-by-detection, Markov Chain Monte Carlo data analysis , motion analysis

Citace

Stanislav Smatana: Analysis of Motion of People by a Stationary Camera, bakalářská práce, Brno, FIT VUT v Brně, 2014

Analysis of Motion of People by a Stationary Camera

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Adama Herouta, Ph.D.

.....
Stanislav Smatana
May 21, 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu doc. Adamu Heroutovi za odbornou pomoc a vedení při realizaci celého projektu.

Contents

Chapter 1

Introduction

In recent decades, computer vision became very vibrant and interesting research field. Moreover, as a result of rapid popularization of mobile devices, nearly every person today has some device with capability of image capture. Therefore, computer vision applications are finding more and more use by common public.

One of the most important applications of this field is detection and tracking of people, which has use in robotics, entertainment, surveillance, care for the eldery and disabled and content-based indexing [9].

Particular aim of this thesis is proposal and implementation of multitarget human tracking-by-detection system with application for security or customer behavior analysis in shops.

Thesis is structured as follows. Chapter 2 will provide brief overview of basic principles and categories of modern multi-target tracking systems. Subsequent chapters are aimed at explanation of fundamental concepts needed to be clarified, in order to understand structure of proposed system. More concretely, chapter 3 will explain methods of object description using feature descriptors and chapter 5 will describe their use together with concept of classifier. Chapter 4 discusses detection and tracking of low-level image features. Finally chapter, 6 will introduce powerful family of sampling methods called Markov Chain Monte Carlo.

After discussion of theoretical concepts, chapters 7 and 8 will follow with proposal and implementation details of multi-target human detection and tracking system. Finally, chapter 9 will conclude the whole thesis with discussion of performance and usability of its implementation.

Chapter 2

Multi-target tracking systems

Essence of multi-target tracking problem is to find eaxct tracks from the set of noisy measurements (e.g. detections). Because the association between these measurements and actual targets is unknown, these kinds of problems are called data association problems [14].

Multi-target tracking systems are categorized based on objective function they are trying to optimize as Heuristic or Bayesian approaches . The former of two typically does not posses any type of explicit objective function. One such example is the greedy nearest-neighbor filter. It associates measurements based on their proximity only, thus resulting in one association per scan. Therefore, this approach breaks down under more difficult conditions . Bayesian approaches are trying to find either Maximum a posteriori (MAP) or the Bayesian estimator solution. While the MAP approaches are outputing solutions with the best encountered probability, the Bayesian estimate approaches minimize posterior expected value of some risk function. Typically the mean squared error [14].

Morover, tracking approaches can be also categorized based on the way they process measurements as single-scan and multi-scan algorithms. Single-scan algorithms use only previously estimated states and current measurements in order to estimate curent states. On the other hand, multi-scan algorithms also incorporate multiple past scans and are able to correct previous estimations in the light of new evidence [14].

Monte Chain Monte carlo data association is example of one such system which is able to estimate both Bayesian estimator and MAP estimate. Moreover, it uses Markov Chain Monte Carlo sampling instead of enumeration of all track configurations. It also incorporates false alarms, missing measurements and the ability to create and destroy tracks. In my thesis, I decided to use approach based on the proposal of Benfold an Reid [12], which closely resembles online multi-scan mcmcda developed by Songhwai, Russel an Sastry [14].

Chapter 3

Feature descriptors for object detection

In order to perform methods of computer vision, relevant information must be extracted from the image. One sort of such information are interesting points , called *keypoints*. Example of those are Good Features to Track produced by Shi-Thomasi [5] corner detector described in section 4.1. These can be then tracked in the image sequence using methods such as Mean Shift, Cam Shift or Kanade-Lucas-Tomasi tracker [6].

Furthermore, these in combination with particular description (e.g. histogram) and neighborhood form *feature descriptor*. However, the descriptor can be also derived for the whole region as in the case of *Histograms of Oriented Gradients* or *Boosted Color Bins*. These are used in combination with classifier (chapter 5) to perform object detection [6].

3.1 Histogram of oriented gradients for object detection

The main idea of using histograms of oriented gradients (in the rest of this thesis referred to as HOG) is that appearance and shape of object can be well characterized by distribution of local intensity gradients. This approach does not require precise knowledge of edge or gradient positions [8].

Overview of the whole process is shown on figure 3.1. Firstly, detector applies gamma or color normalization to input image. However, this step proved to have only modest effect on overall detector performance. Next, it calculates image gradients. In case of color images, calculation is performed on all channels for each pixel and gradient with largest norm is selected. Newly created gradient channel is then divided into spatial regions of chosen dimensions called cells with either rectangular or radial character. For each cell, the detector constructs a histogram of gradient orientations with evenly spaced orientation bins. Histogram vote is a function of gradient magnitude and is bilinearly interpolated between neighboring bin centers to reduce aliasing. Furthermore, detector groups cells into larger spatial regions called blocks which are used for local contrast normalization. This turns out to be cruicial for good performance. Finally, a vector of all components of the normalized cell responses from all of the blocks in the detection window is created . Resulting vectors can be used to train Support Vector Machine or other form of classifier [8].

HOG features provided substantial gains over intensity-based features. After their popularization by Dalal and Triggs [8], they have been used in nearly all modern pedestrian detectors in some form [9].

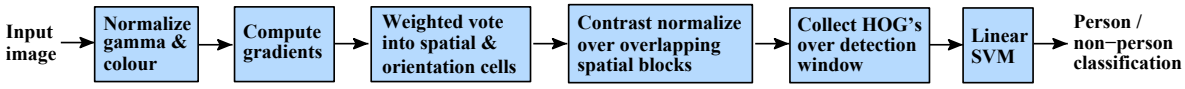


Figure 3.1: Figure shows overview of HOG feature creation and training (taken from [8])

3.2 Boosted color bin features

In the domain of object tracking , traditional features dedepending on spatial configuration (e.g. Histograms of Oriented Gradients, Haar Wavelets) fail, because moving objects may exhibit improptu changes of shape. In these cases, color histogram provides more robust information about object appearance. However, computation of a full 3D RGB color histogram is very time-consuming [18].

Boosted color bin features address this issue by using only a set of 13 1D color histograms. Values of every pixel in local window are projected onto 13 lines passing through point $[128, 128, 128]$ in RGB color space (figure 3.2). Each line is segmented into 8 regions representing bins of its histogram. After every projection, histogram bin corresponding to it's line segment is incremented. Concatenation of normalized histograms of projected values from each line then forms final feature vector. This can still be time consuming operation, but with help of special data structure called *integral histogram* , feature vector can be calculated in constant time [18].

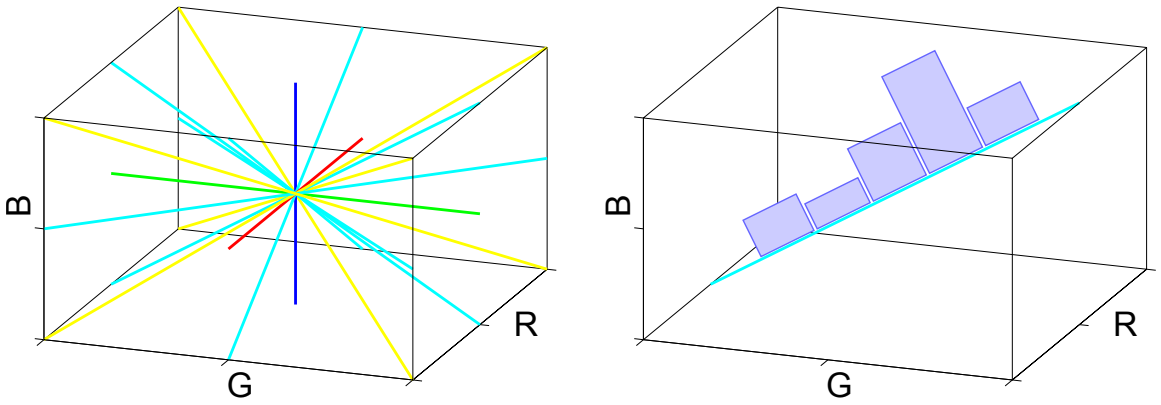


Figure 3.2: Example of lines in RGB space and 1D color histogram (taken from [18])

3.2.1 Integral histogram representation

Integral histogram is data structure used to overcome major performance bottleneck in computer vision applications caused by frequent histogram calculation. Using this approach, histogram for the desired region can be calculated using only small set of simple arithmetic operations [10].

In plain words, value of integral color histogram at given point of two dimensional image is the histogram between this point and image origin. If the origin is the top left corner, value of integral histogram at bottom right corner is the histogram of whole image.

Construction of data structure is efficiently performed based on idea of *propagation*. Using it, integral histogram $H(\mathbf{x}^p, b)$ at the p^{th} order along a sequence of points $\mathbf{x^0}, \mathbf{x^1}, \ldots, \mathbf{x^p}$

can be formally expressed as

$$H(\mathbf{x}^p, b) = \bigcup_{j=0}^{p} Q(f(\mathbf{x}^j)) \tag{3.1}$$

where function f is mapping from d-dimensional Cartesian space to k-dimensional tensor and where $Q(.)$ denotes corresponding bin of current point. The union operator $\cup$ is defined as follows: value of the bin b of $H(\mathbf{x}^p, b)$ is equal to sum of the previously visited point's bin values. This means that $H(\mathbf{x}^p, b)$ is the histogram of the region between the origin and current point (figure 3.3 ilustrates this principle). As a result, integral histogram can be defined recursively:

$$\begin{aligned} H(x^j, b) &= H(x^{j-1}, b) \cup Q(f(x^j)) \\ H(0, b) &= 0 \end{aligned}$$

where $H(0, b) = 0$ means that all bins are 0 at the origin [10].

Construction of a histogram can be achieved by various types of propagation and scanning. In this approach, I use a method named *wavefront scan*, in which, at each step, current value is obtained from the values of three neighbors together with increasing bin corresponding to processed pixel. This principle is ilustrated on figure 3.3.

Histogram of the desired region is calculated using operation named *intersection*. For general integral histogram of d dimensions, it is defined as follows:

$$h(T, b) = \sum_{r=0}^{d} (-1)^r \sum_{l=1}^{C_r^d} H(\mathbf{x}_l^r, b).$$

in $N_1 \times N_2$ gray level image this becomes:

$$h(T, b) = H(p_1^+, p_2^+, b) - H(p_1^-, p_2^+, b) - H(p_1^+, p_2^-, b) + H(p_1^-, p_2^-, b)$$

and propagation can be written as:

$$H(x_1, x_2, b) = H(x_1 - 1, x_2, b) + H(x_1, x_2 - 1, b) - H(x_1 - 1, x_2 - 1, b) + Q(f(x_1, x_2))$$

[10] [13].

In conclusion, integral histogram allows for region histogram calculation to be performed in constant time. Construction of structure itself is completed in linear time. Thus, greatly increasing performance of application relying on histogram calculation [10].

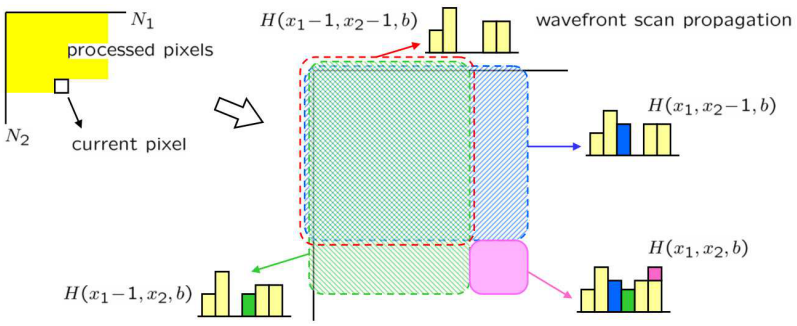


Figure 3.3: Wavefront propagation in integral histogram (taken from [10])

Chapter 4

Interest points detection and tracking

4.1 Shi-Tomasi Corner Detector

The main purpose of Shi-Tomasi corner detector is to find image locations (corners), which are suitable for establishing correspondencies with other images. It is based on the detector by Harris [4] which extends earlier work of Moravec [6].

Main intuition behind this method is that easily localizable image patches exhibit large contrast changes, whereas flat areas have uniform intensity and are nearly impossible to find. Moravec detector exploits this idea by considering a local window in image, and determining average changes of intensity that result from it's shifting in four general directions. Based on this approach, three situation can be encountered:

- Shifts result in small change of intensity - windowed region is flat;
- Shifts result in small change of intensity in one direction - window is on edge region;
- All shifts result in large change of intensity- windowed patch is corner or isolated point.

Change of intensity can be mathematically expressed as

$$E(x,y) = \sum_{u,v} w_{u,v} |I(x+u, y+v) - I(u,v)|^2$$

where $I(x,y)$ is value of image intensity at given point and w is the window function. In case of the Moravec detector, it has value of one in specified rectangular region and zero elswhere. Detector then simply looks for local maxima in $min(E)$. However, this approach suffers from multiple problems, which are addressed by Harris detector [4].

Firstly, considering shifts only in 4 general directions leads to anisotropic response. Nevertheless, all possible small shifts can be evaluated using analytic expansion about the shift origin. For small shifts this leads to the formula:

$$E(x,y) = Ax^2 + 2Cxy + By^2$$

where

$$\begin{aligned} A &= X^2 \otimes w \\ B &= Y^2 \otimes w \\ C &= (XY) \otimes w \\ X &= I \otimes (-1, 0, 1) \approx \partial I / \partial x \\ Y &= I \otimes (-1, 0, 1)^T \approx \partial I / \partial y \end{aligned}$$

Another problem is noisiness of response because of binary rectangular window. Harris and Stephens address this problem using the Gaussian window instead:

$$w_{u,v} = exp(\frac{u^2 + v^2}{2\sigma^2})$$

Lastly, the Moravec operator responds too readily to edges. This drawback is caused by taking into account only the minimum of E and can be solved by making use of the variation of E with the direction of shift. Thus, leading to problem formulation for small shifts in terms of 2 $\times$ 2 symmetric matrix M:

$$E(x, y) = (x, y)M(x, y)^T$$

closely related to the local autocorrelation function [4].

Harris and Stephens [4] define detector response in terms of eignevalues α and β of matrix M as:

$$R = \alpha\beta - k(\alpha + \beta)^2. \tag{4.1}$$

For negative values of R, area is considered to be flat, for zero edge, and for positive values a corner candidate. Candidate is accepted as corner if it's response is an 8-way local maximum [4].

Shi-Thomasi in their research paper Good Features to Track [5] further extended this principle by changing response function to simple thresholding. Only windows for which $min(\alpha, \beta)$ is greater then chosen threshold λ are accepted as corner regions.

4.2 Kanade-Lucas-Tomasi Feature Tracker

Features such as those mentioned in previous section, can be tracked from frame to frame in an image sequence. In my work, I use the Kanade-Lucasi-Tomasi feature tracker together with Shi-Tomasi corners to estimate motion of people.

Basic idea of this tracking approach is that images taken in near instants are usually strongly related to each other. Following term expresses this correlation:

$$I(x, y, t + \tau) = I(x - \zeta, y - \eta, t)$$

I represents intensity value in image stream as a function of pixel position x, y and time t. In plain words, image at time $t + \tau$ is obtained by moving every point in image at time t by suitable *displacement*. However, this criterion may be violated due to occlusion or when tracked feature leaves image space [3].

Moreover, tracking cannot be applied to single pixel only, because it is unlikely to have very distinctive brightness with respect to its neigbhours. As a result, tracker uses

the whole window of pixels. This may lead to the use of rich model with great number of parameters, although only small window with two parameters (displacement vector) is sufficient. Considering this, the model can me be redefined as follows:

$$J(\mathbf{x}) = I(\mathbf{x} - \mathbf{d}) + n(\mathbf{x})$$

where n is noise, $J(\mathbf{x}) = I(x, y, t + \tau)$ and $I(\mathbf{x} - \mathbf{d}) = I(x - \zeta, y - \eta, t)$ [3, 16].

Displacement vector is then chosen to minimize residue error defined as integral over window W:

$$\varepsilon = \int [I(\mathbf{x} - \mathbf{d}) - J(\mathbf{x})]^2 w d\mathbf{x}$$

where w is a weighting function. [5]. Term $I(\mathbf{x} - \mathbf{d})$ can be approximated using it's Taylor expansion leading to system of linear equations defined as $G\mathbf{d} = \mathbf{e}$, where G is 2×2 matrix define as:

$$G = \left[\frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \right]$$

Computation of vector $\mathbf{e}$ requires computation of difference between frames. Vector d is result of tracking between two frames [3, 16].

4.3 Kalman filter

The Kalman filter is a recursive solution to the discrete data linear filtering problem. More concretely, it tries to estimate the state $x \in \mathcal{R}^n$ of a discrete-time controlled process governed by equations

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

with measurement

$$\begin{aligned} z_k &= Hx_k + v_k \\ z &\in \mathcal{R}^m \end{aligned}$$

where random variables w_k and v_k represent process and measurement noise with normal probability distribution. Matrix A relates a current state to a past state and matrix H relates the current state to the measurement. Optionally, filter may use the control input u together with its relationship matrix B [17].

From algorithmic point of view, the filter is executed in two steps. First is the time update or prediction, second is the measurement update or correction. Together, they form a predictor-corrector relationship (figure 4.1 shows an overview of the algorithm) [17].

The update step projects both current state and error covariance into the future. The predicted state is calulated usin equation 4.2, without the noise term and the error covariance is projected as follows:

$$P_k^- = AP_{k-1}A^T + Q$$

where A is matrix from equation 4.2 and Q is the process noise covariance. While the measurement noise covariance R from equation 4.2 is usually measurable, the process noise covariance Q is generally difficult to estimate [17].

The measurement update step, performs correction of state based on a new measurement. Its first task is to calculate the *kalman gain* K_k using following equation:

$$K_k = P_k^- H^T (HP_k^- + R)^{-1} \tag{4.3}$$

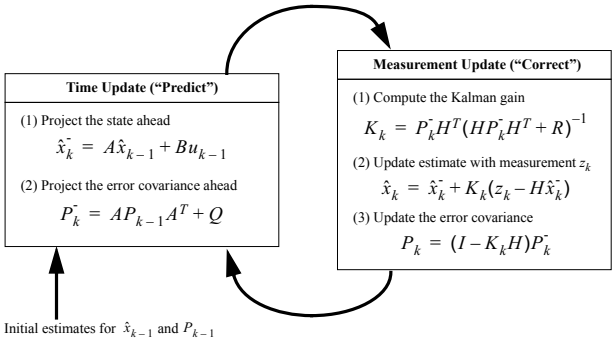


Figure 4.1: Overview of kalman filter algorithm

The main purpose of the Kalman gain is to correct predicted state in the light of the newly acquired measurement. Its calculation is expressed as

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{4.4}$$

and using it, both the error covariance and the state are corrected as follows [17]:

$$x_k = x_k^- + K_k(z_k - Hx_k^-) \tag{4.5}$$
$$P_k = (I - K_k H)P_k^- \tag{4.6}$$

After each correction and prediction, resulting state is used as prior state of new iterations. This recursive nature of the filter makes it very practical for implementation [17].

Chapter 5

Methods of data classification

The task of classification is to assign input data to a number of classes (e.g. distinguish between people and other objects). This association is based on classification rule learned by observing past data [1]. Examples of such classification methods in the domain of supervised learning are *Support Vector Machines* (SVM) and *Adaptive Boosting* (Adaboost). The former of mentioned was used by Dalal and Triggs [8] in combination with Histogram of Oriented Gradients for pedestrian detection. Latter was used by Wei et al. [18] for offline color tracking objects. I used both of these principles in my thesis, therefore they will be subject to further discussion.

5.1 Support vector machines

Support vector machines (refered to as SVMs) are a class of algorithms for classification that use the idea of *kernel substitution*. In the course of their existence, they were successfully applied to a number of applications ranging from face identification or text categorization to engine knock detection [7].

Let us consider binary classification task with datapoints $x_i (i = 1, \ldots, m)$, labels $y_i = \pm 1$ and classification function $f(x) = sign(w \cdot x - b)$, where vector w describes orientation of the discriminant plane separating both classes and scalar b determines its offset from origin. Best among planes separating these classes can be characterized as „furthest“ from both of them. Furthermore, a plane with such properties bisects closest points in the convex hulls of classes (ilustrated in figure 5.1). These points can be found by solving following quadratic problem [7]:

$$\begin{aligned} min_\alpha \quad & \tfrac{1}{2} ||c - d||^2 \\ & c = \sum_{y_i \in Class1} \alpha_i x_i \qquad d = \sum_{y_i \in Class-1} \alpha_i x_i \\ s.t. \quad & \sum_{y_i \in Class1} \alpha_i = 1 \qquad \sum_{y_i \in Class-1} \alpha_i = 1 \\ & \alpha_i \geq 0 \qquad i = 1, \ldots, m \end{aligned} \tag{5.1}$$

More importantly, an identical result can be achieved with use of *supporting planes*. Plane supports class if all points in that class are in one side of that plane. One such plane is selected for each class and then maximization of margin between them is performed, until supporting planes lie on few points (*support vectors*) from each class. It can be expressed by the term $\gamma = \frac{2}{||w||_2}$. Its maximization is equal to minimization of term $\frac{||w||_2}{2}$, which leads to the following quadratic program [7]:

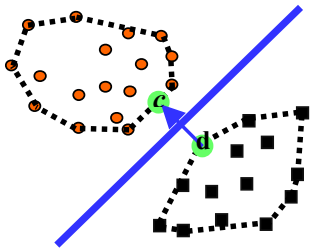


Figure 5.1: Figure shows two classes, their convex hulls and closest points c and d (taken from [7])

$$\begin{array}{lll} \min & & _{w,b}\frac{1}{2}||w||^2 \\ s.t. & w \cdot x_i \geq b+1 & y_i \in Class1 \\ & w \cdot x_i \leq b+1 y_i & \in Class-1 \end{array} \tag{5.2}$$

Previous example was a case of classes which are linearly well-separable. However, SVMs can be also used to solve non-linear classification problems. This can be achieved simply by substituting dot product with a kernel function evaluation . Dalal and Triggs [8] used SVM with either linear or gaussian kernel in combination with HOG descriptors for pedestrian detector [7].

$\theta(u)$	$K(u,v)$
Degree d polynomial	$(u \cdot v+1)^d$
Radial Basis Function Machine	$exp\left(-\frac{\|\|u-v\|\|^2}{2\sigma}\right)$
Two-Layer Neural Network	$sigmoid(\eta(u \cdot v)+c)$

Figure 5.2: Examples of kernel function, taken from [7]

5.2 Adaptive Boosting classifier

In general, boosting is a method to improve any learning algorithm by combining the number of weak classifiers into one strong, final classifier. The only requirement being, that weak classifiers are a little bit better than random guessing . It should be noted that AdaBoost itself is a general framework ant the both weak classifiers and their learning method may differ from case to case [19].

One of the central concepts of Adaptive Boosting are training sample weights. At first these are equal for all samples. But, in subsequent iterations, weights are changed in order to direct learning towards harder training examples, thus, giving more expressive power to a weak learning algorithm. Weak learning algorithm provided with weight distribution generates a weak classifier with hypothesis of classification h_t. This is used to calculate the error called *pseudo-loss* with notation ε (equation 5.3).This is in turn used to modify weights of samples. The whole process is then repeated for a number of iterations . After each iteration the algorithm generates a new weak classifier. Number of these than form final strong classifier [13].

$$\varepsilon_t = \frac{1}{2}\sum_{(i,y)\in B} D_t(i,y)(1-h_t(x_i,y_i)+h_t(x_i,y)) \tag{5.3}$$

Input: sequence of m examples $[(x_1, y_1), \ldots, (x_m, y_m)]$ with labels $y_i \in Y = 1, \ldots, k$
Weak learning algorithm **WeakLearn**,T - number of iterations
Let B = $(i, y) : i \in 1, \ldots, m, y \neq y_i$;
Initialize $D_1(i, y) = 1/|B| for (i, y) \in B$.
for $t = 1, 2, \ldots, T$ **do**
*Call **WeakLearn**, providing it with mislabel distribution D_t;*
Get back a hypothesis $h_t : X \times Y[0, 1]$;
Calculate preudo-loss of h_t: $\varepsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$;
Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ $UpdateD_t$: $D_{t+1}(i, y) = \frac{D_t(i,y)}{Z_t} \cdot \beta_t^{(1/2)(1+h_t(x_i,y_i)-h_t(x_i,y))}$
where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).
end
Output: *the hypothesis:* $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(x, y)$

Algorithm 1: Adaboost.M2 algorithm as proposed by Yoav Feund and Robert E. Schapire [19]

Specifically, I use AdaBoost together with color bin features (described in section 3.2 on page 5) for person representation as described by Yichen Wei et al. [18]. In this approach, the training set is represented as $m \times n$ matrix where every row represents an observed sample (color histogram of image area) and every column is a histogram bin. Weak learning is performed by sorting each column and finding in it threshold, which best separates positive and negative samples. The best among them is than used as a weak classifier. As a result, weak classifier is defined as

$$h(x) = \begin{cases} 1 & \text{if} sx < x\theta \\ 0 & \text{otherwise} \end{cases}$$

where x is a single color bin feature from particular column, $s \in \{-1, 1\}$ is the classifier polarity. And θ denotes it's threshold [18]. The response of final strong classifier is obtained by following formula:

$$h^{strong}(x) = \begin{cases} 1 & \sum_{t=1}^{T} a_t h_t(x) \geq \sum_{t=1}^{T} a_t \\ 0 & otherwise \end{cases}$$

Any number of weak classifiers with error lower than 0.5 can be used in the final strong classifier. Errors greater than 0.5 would cause divergence of algorithm, therefore evaluation should be stopped after encountering first such classifier [13].

Chapter 6

Monte Carlo Markov Chain Data Association methods

Statistical inference based on most of the practical probabilistic models is out of current computational capabilities, therefore it must be supplemented by reasonable approximation. For the purpose of this thesis, one family of methods, called *Monte Carlo*, based on numerical sampling is of particular interest. Together with *Markov Chains*, they form a very general and powerful framework called *Markov Chain Monte Carlo* able to sample from large class of distributions [2].

6.1 Markov chains

The Markov chain of first-order is defined as a series of random variables $\mathbf{z^{(1)}}, \ldots, \mathbf{z^{(M)}}$ that hold *markov property* described by the equation 6.1. In other words, discrete-time markov chain is a stochastic process in which each new state $\mathbf{z}^{n+1}$ depends only on state $\mathbf{z}^n$ and not on states $\mathbf{z}_k$ for $k \leq n$. Practical example of this principle might be a piece of music, where every played note depends just on it's predecessor [11].

$$p(\mathbf{z^{(m+1)}}|\mathbf{z^{(1)}}, \ldots, \mathbf{z^{(m)}}) = p(\mathbf{z^{(m+1)}}|\mathbf{z^{(m)}}) \tag{6.1}$$

Markov chain can be also specified by probability distribution of initial variable $\mathbf{p(z^{(0)})}$ together with transition probabilities T_m for subsequent variables defined as $T_m(\mathbf{p(z^{(m)})}, \mathbf{p(z^{(m+1)})}) \equiv p(\mathbf{z^{(m+1)}}|\mathbf{z^{(m)}})$ in form of transition matrix [11].

Also, Markov chain is said to have a *limiting distribution* if the limit

$$lim_{n\to\infty}\mathcal{P}(X_n = j|X_0 = i)$$

exists for all i, j in state space $\mathcal{S}$ and form a probability distribution on S. Furthermore, this distribution is said to be *stationary* if each step in Markov Chain leaves it so. Also, it is invariant with respect to matrix P meaning that $\pi = \pi P$. This means that if the chain is started with stationary distribution it will preserve it in all subsequent states, whereas in case of limiting distribution it will reach it after an infinite number of steps [11]. Sufficient, but not a necessary condition for distribution to be invariant is called *detailed balance* (equation 6.2). [2].

$$p^\star(\mathbf{z})T(\mathbf{z}, \mathbf{z}') = p^\star(\mathbf{z}')T(\mathbf{z}', \mathbf{z}) \tag{6.2}$$

In MCMC methods, Markov Chains are constructed for purpose of sampling from probability distribution. In order to be used for this purpose, it must hold property called

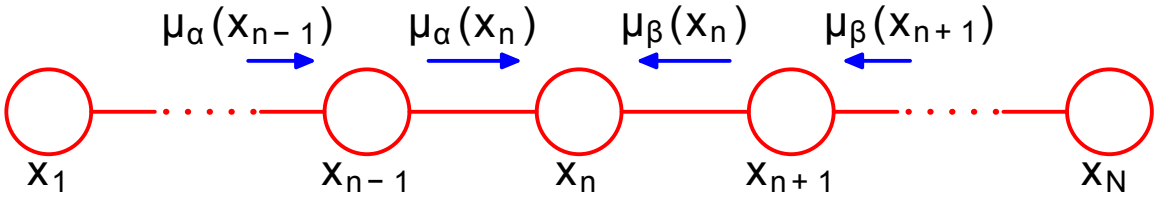


Figure 6.1: Markov chain represented by directed graph, (taken from [2])

ergodicity, which means that for $m \to \infty$, the distribution $p(\mathbf{z}^{(m)})$ converges to required invariant distribution $p^{\star}(\mathbf{z})$, without dependence on choice of initial distribution $p(\mathbf{z}^{(0)})$. It is then called *equilibrium* distribution, and is unique for a given ergodic Markov Chain [2].

6.2 Metropolis-Hastings algorithm

Metropolis-Hastings algorithm is the most influential of Markov chain Monte Carlo (MCMC) family of algorithms [15]. The main purpose of these methods is to draw samples from some non-trivial probability distribution. Typically, this is done in order to evaluate expectation of function with respect to this distribution. Thus, substituting integration

$$\mathcal{E}[f] = \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

for approximation by finite sum

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$$

[2].

In general, MCMC generates samples from distribution π on a space ω by constructing a Markov chain $\mathcal{M}$ with states $w \in \omega$ and stationary distribution $\pi(w)$ [15].

Let us consider the state ω of markov chain $\mathcal{M}$. In each step Metropolis-Hastings algorithm proposes new state ω' following proposal distribution $q(\omega, \omega')$. This distribution should be simple enough, so that it can be sampled directly. New state is then accepted with probability:

$$A(\omega, \omega') = min\left(1, \frac{\pi(\omega')q(\omega', \omega)}{\pi(\omega)q(\omega, \omega')}\right) \tag{6.3}$$

otherwise the sampler stays at state ω. This process is repeated for chosen number of iterations. It's convergence strongly depends on choice of proposal distribution. Furthermore, it should be noted, that term $\frac{q(\omega',\omega)}{q(\omega',\omega)}$ was introduced in order to allow sampling with non-symmetric proposal distribution. Without it, it is no longer Metropolis-Hastings algorithm, but its earlier version called *Metropolis* algorithm.

Moreover it can be easily shown, that a Markov Chain produced by Metropolis-Hastings algorithm holds detailed balance. Using equation 6.2 and 6.3:

$$\begin{aligned} p(\mathbf{z})q_k(\mathbf{z}|\mathbf{z}')A_k(\mathbf{z}', \mathbf{z}) &= min(p(\mathbf{z}), q_k(\mathbf{z}|\mathbf{z}'), p(\mathbf{z}')q_k(\mathbf{z}'|\mathbf{z})) \\ &= min(p(\mathbf{z}'), q_k(\mathbf{z}'|\mathbf{z}), p(\mathbf{z})q_k(\mathbf{z}|\mathbf{z}')) \\ &= p(\mathbf{z}')q_k(\mathbf{z}'|\mathbf{z})A_k(\mathbf{z}, \mathbf{z}') \end{aligned}$$

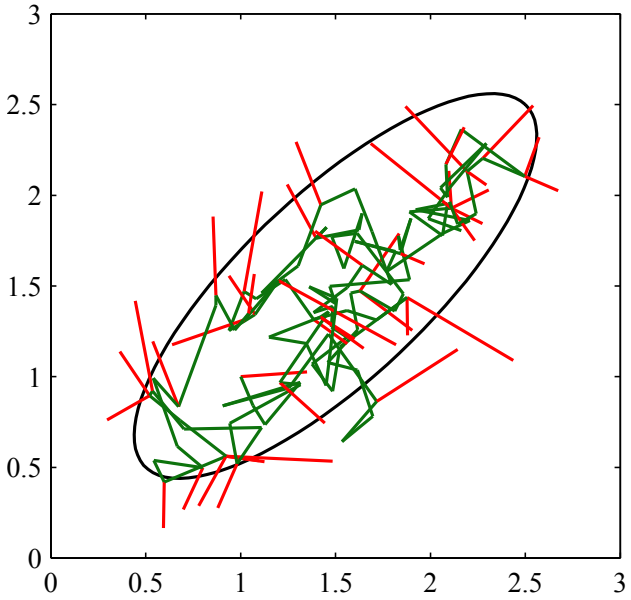


Figure 6.2: Ilustration of Metropolis-Hastings algorithm sampling from a Gaussian distribution. Green lines are accepted steps, red are rejected ones (taken from [2]).

MCMC methods have been used for complex integration, counting and combinatorial optimization problems, sometimes being the only family of methods able to find good approximate solution in polynomial time [15].

Chapter 7

Proposal of pedestrian detection and tracking system

Figure 7 demonstrates the overall architecture of proposed pedestrian detection and tracking system. It consist of detection, Adaptive Boost training and MCMCDA unit. This approach is based on a proposal of Ben Benfold and Ian Reid [12]. However, I decided to use sightly different probability calculation method (described in section 7.3.1 on page 19), different detection approach and to incorporate the AdaBoost classifier into hypothesis probability calculation.

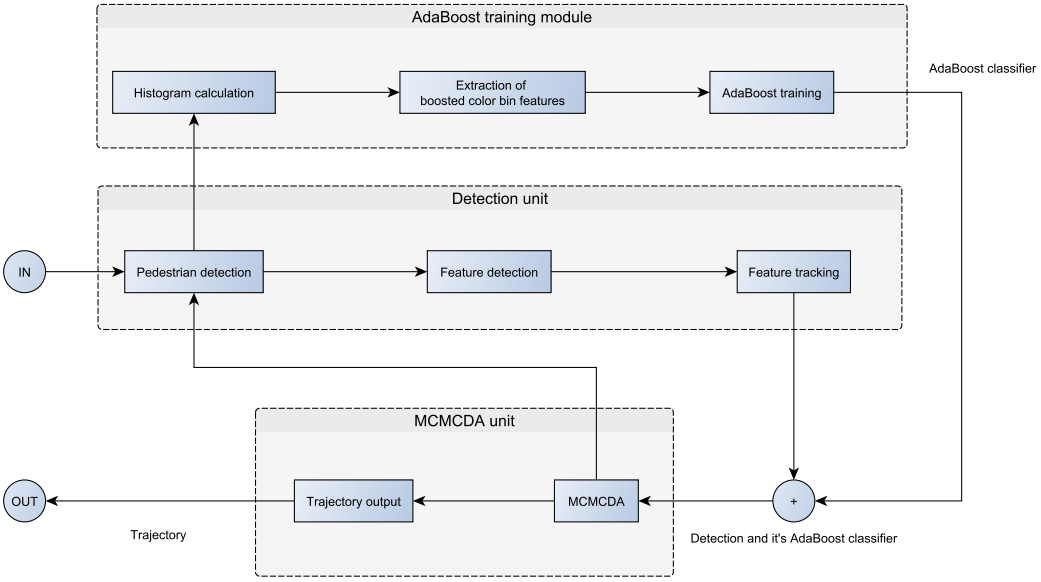


Figure 7.1: Overall architecture of detection and tracking system

7.1 Detection unit

The detection unit detects people in frame using a combination of Histograms of Oriented Gradients with Support Vector Machine. This method could be substitued by any other of existing approaches, however, accurate detections are of cruicial importance for system's

performance. Moreover, in this approach, detections are not performed in every frame, but rather once in a chosen time interval. As a consequence, motion estimates are needed to ensure correct data association.

These should be calculated from frame-to-frame tracking of some low-level features. I decided to use Good Features To Track as described by Shi and Tomasi [5] in combination with Kanade-Lucas-Tomasi Feature Tracker. Apart from motion, I also experiment with using boosted color bin features as an appearance descriptor of a particular person.

Figure 7.1 demonstrates the function of the whole detection system. Firstly, at the beginning of detection interval, pedestrians in frame are detected. Next, low-level feature detection is performed in all detected bounding boxes. Apart from that, detection unit also performs foreground-background separation. This is done, in order to minimalize chance of detecting low-level features on background instead of moving person. Tracking is performed into both future and to past for a set amount of time, thus resulting in two sets of motion estimates per detection.

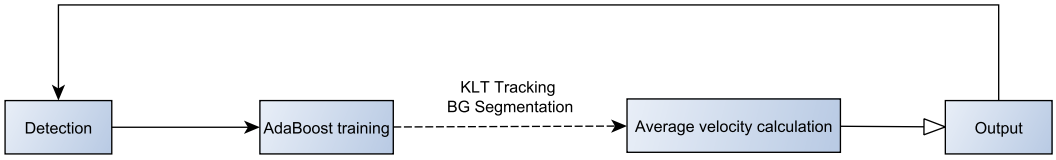


Figure 7.2: Diagram of detection unit

7.2 Adaptive Boosting classifier training unit

This unit is responsible for boosted bin feature extraction and training of AdaBoost classifiers for individual detections. Positive samples for training are obtained by caluclating histograms from detection area and its various little scalings and shifts. In order to better distinguish a particular person, histograms of other detections are taken as negative samples. Also, histograms from a number of randomly chosen rectangular areas are included to further differentiate person in contrast to his background. Trained classifier is than used for trajectory probability calculation in the AdaBoost probability module .

7.3 Monte Carlo Markov Chain data association unit

Central part of this unit is the Metropolis-Hastings algorithm. However, it is not used to estimate expectation of some function, but to find track configuration with highest probability. Initial state is obtained by assembling detections into tracks based on their proximity. Apart from detections, the track is also assigned either to class of false positives or true positives. In each iteration of algorithm, the unit generates new track configuration by executing one of three moves as proposed by Benfold and Reid in their paper Stable Multi-Target Tracking in Real-Time Surevillance Video [12]. First type of move is called swap and involves selecting random detection in track and swapping it with detection from another track. If there is no detection at equal point in second track,it is simply moved. Second type of move (switch) , moves all detections starting at random point from one track to other and vice-versa. Last type of move toggles class of randomly selected track between

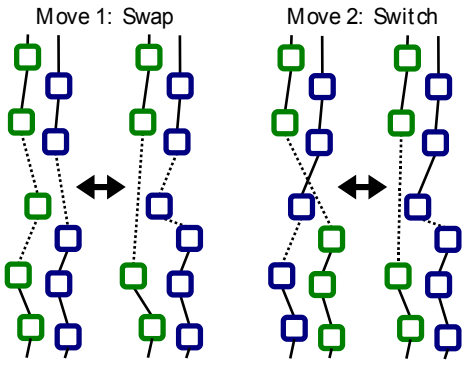


Figure 7.3: Moves used in making new state proposals (taken from [12])

false positive and true positive. Both tracks and beginning points of moves are randomly selected using uniform probability distribution. To allow track creation, one additional empty track is provided.

After each iteration, the algorithm needs to evaluate current hypothesis probability. Ian and Benfold [12] use following probability formula:

$$p(D|H_i) = \prod_{T_j \in H_i} \left[p(d_1^j|c_j) \prod_{d_n^j \in T_j - \ d_1^j} p(d_n^j|d_{n-1}^j, c_j) \right]$$

where H_i denotes tracking hypothesis d_n detection and c_j class of track T_j. However, the probability of a track can be evaluated based on numerous criteria. I use two main probability modules: module based on motion similar to that proposed by and one based on AdaBoost Classifier. Finally, when number of iterations reaches limit, algorithm outputs best tracking hypothesis encountered.

As a whole, the unit operates in a fashion similar to online MCMCDA with time window proposed by Oh, Russel and Sastry [15]. System processes incoming detections from detection unit in time window of particular size as ilustrated on figure 7.3. Firstly, window is filled with detections from first n detection intervals. After that, unit performs initial MCMCDA. In successive steps, MCMCDA follows every addition of new measurements from single detection interval along with advancement of window by one set of measurements. However, probability is calculated considering additional n measurements in the past, thus forming full window of size $2n$ consisting of n movable and n unmovable sets of detections.

After all detection in particular track leave space of temporal window, it's tracking is completed and it can be sent to output.

7.3.1 Probability modules

After every step of Metropolis-Hastings algorithm, probability modules calculate the probability of the new hypothesis. I propose two modules - one based on movement and second based on appearance.

First module is based on one proposed by Benfold and Ian [12]. It calculates probability based on position, klt motion estimates and average velocity of subsequent detections ,considering linear motion model of person. Although, this is not exact, it is sufficient

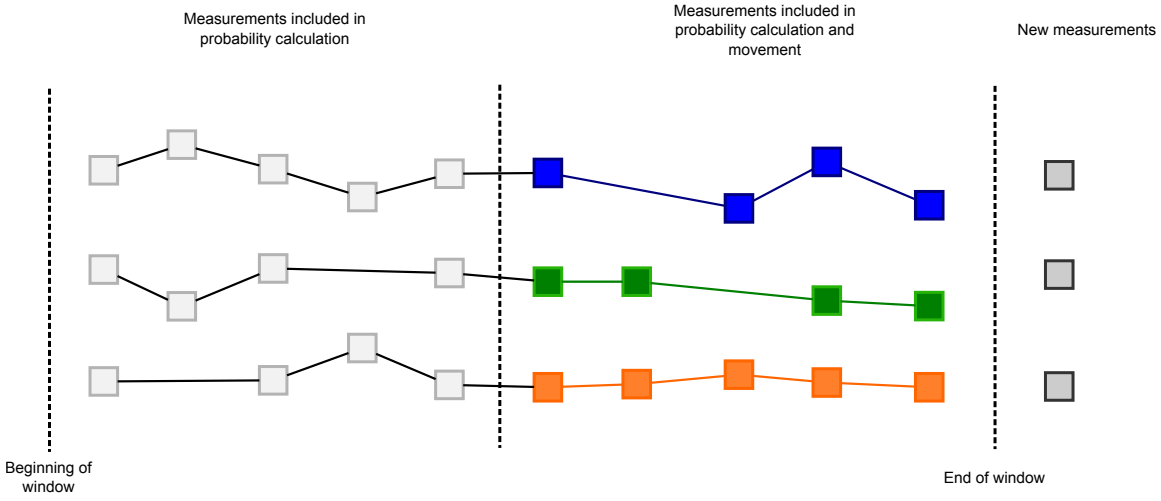


Figure 7.4: Diagram of sliding window

approximation for short distances. Having two subsequent detections d_{n-1} and d_n in track t motion model is represented as

$$x_p = x_{n-1} + \delta_t v_p$$

where x_{n-1} denotes position of detection d_{n-1} and δ_t is the time difference between detections. However, KLT tracking provides much more accurate prediction of movement. Therefore , for each motion estimate $\mathbf{y}$ unit calculates posterior distribution over locations using calculation equivalent to a step of kalman filter:

$$x_y = x_{n-1} + \Sigma_p(\Sigma_p + \delta_t\Sigma_{klt})^{-1}(x_{n-1} + y - x_p)$$

where Σ_p is error caused by unknown accelerations, and Σ_{klt} is rate in which klt accumulates random error. The unit uses this estimate to calculate probability of location x_n as

$$x_n|x_{n-1}, \mathbf{y}, c_{ped} \sim \alpha^{\delta_t}\frac{1}{|\mathbf{y}|}\sum_{y\in\mathbf{y}}\mathcal{N}(x_y, \Sigma_y + 2\Sigma_d) + (1-\alpha^{\delta_t})\mathcal{N}(x_p, 2\Sigma_d)$$

where c_{ped} is class of track and parameter α is included for purpose of modeling failure of KLT tracking. The product of probabilities of all subsequent pairs of tracks is than final result.

Second module uses AdaBoost classifiers produced in AdaBoost unit to calculate probability of a given track. For each subsequent pair of detections d_{n-1} and d_n classifier of d_{n-1} is used to classify d_n based on it's color bin histogram. Probability is than calculated ratio of number of correctly classified pairs to number of all pairs.

Chapter 8

Implementation of proposed system

The whole system is implemented using C++ 11, OpenCV and QT libraries. Its architecture is based exactly on proposal shown on figure 7. However, instead of implementing all units in serial manner, I decided to use the parallel model and run parts of the system in different threads. Morover, I also use OpenCV's graphics card accelerated modules. This led to significant performance gain.

Detection unit

The detection unit uses OpenCV's HOG detector to detect people in video every second. It's output are objects of type `Detection`. These are an integral part of the system living throughout the whole process. Apart from position of person, they also contain Kanade-Lucas-Tomasi tracking motion estimates and Adaptive Boost classifier for future reidentification.

Figure 8.1: Structure of Detection, the central class of whole system

The whole process of tracking and detection is controled by an object of class `Tracker` which is the central part of whole unit. It uses an object of class `KLTTracking` to perform

tracking on all detections. Tracking is performed to both future and past. While past tracking is completed at once, future tracking is executed gradually. Because tracking information is stored in detection objects, it may seem logical for them to also control it's execution. However, it's separation allows for performing long-run klt tracking even after detection leaves this unit and possible future parallelization of klt tracking. Apart from the tracking itself, this module uses background-foreground subtraction accelerated by OpenCL, to ensure tracking of foreground corners only.

After the tracking of detection for amount of four seconds into past and one into future, it is sent to MCMCDA unit. Apart, from detection creation, detection unit is also responsible for execution of AdaBoost training thread.

AdaBoost thread is also responsible for integral histogram calculation. It is performed by the `IntegralHistogram` object. Because initial propagation of the histogram is a very time-consuming operation, I used pointer arithmetics and decided to allocate whole histogram as one continous chunk of memomry.

After calculation is finished, the unit extracts histograms of all detected regions. Every training run uses histogram of actual detection as a positive sample and histograms of other detections as negative samples. Calculation is performed by AdaBoost class.

Monte Carlo Markov Chain Data Association unit

The Metropolis-Hasting algorithm is implemented in the function `metropolis`, which takes object of class derived from an abstract class `BaseHypothesis`. There is a number of methods that derived classes must implement, namely:

- `probability()` - ability to calculate the probability of itself
- `move()` - ability to performe some move resulting in state change
- `undo()` - reverts to the state before move
- `commit()` - makes new state permanent
- `proposal_coef()` - returns $\frac{q(\omega',\omega)}{q(\omega,\omega')}$, term for Metropolis-Hastings
- `markbest()` - marks the current state as best
- `reverttobest()` - returns to the last state marked as best

Because of this, it is general and can be effectively reused for different tasks.

The most important derived class of `BaseHypothesis` is class `Hypothesis`, which represents the tracking hypothesis (figure 8.2 shows it's structure). It stores a list of trajectories along with associated detections. In every step of MCMCDA, the hypothesis invokes the `Mover` object, that executes one or more moves. Each move is implemented as a function that takes three parameters - two `t_trajectory` objects and frame number from which a move should start. Selection of them is the responsibility of `MoveSelector` and trajectory selection for moves is performed by `TrajectorySelector`. In addition, their execution does not lead to copying of the hypothesis track configuration. The Hypothesis object maintains internally stack of performed moves, therefore undo operation then only executes them with swaped trajectory arguments. After moves are finished, the unit calculates probability. For this purpose it uses the class `ProbabilityCalculator` along with probability modules. Probability modules to be used by the unit are defined in configuration xml file that is

loaded when the application starts. Apart from proposed probability calculation modules, I decided to create another one based of detection background-foreground ratio to further distinguish false positives from true positives. All probability calculations are performed in the logarithmic domain, because with standard probabilities, system experienced numerical instability. Moreover, probability calculations have to be as fast as possible because they are calculated numerous times in every round of Metropolis-Hastings. To allow further fine-tuning of probability modules, the appliation also supports special configuration gui mode invoked by `-hgui` commandline argument. It allows to interactively inspect and configure process of the mcmcda sampling.

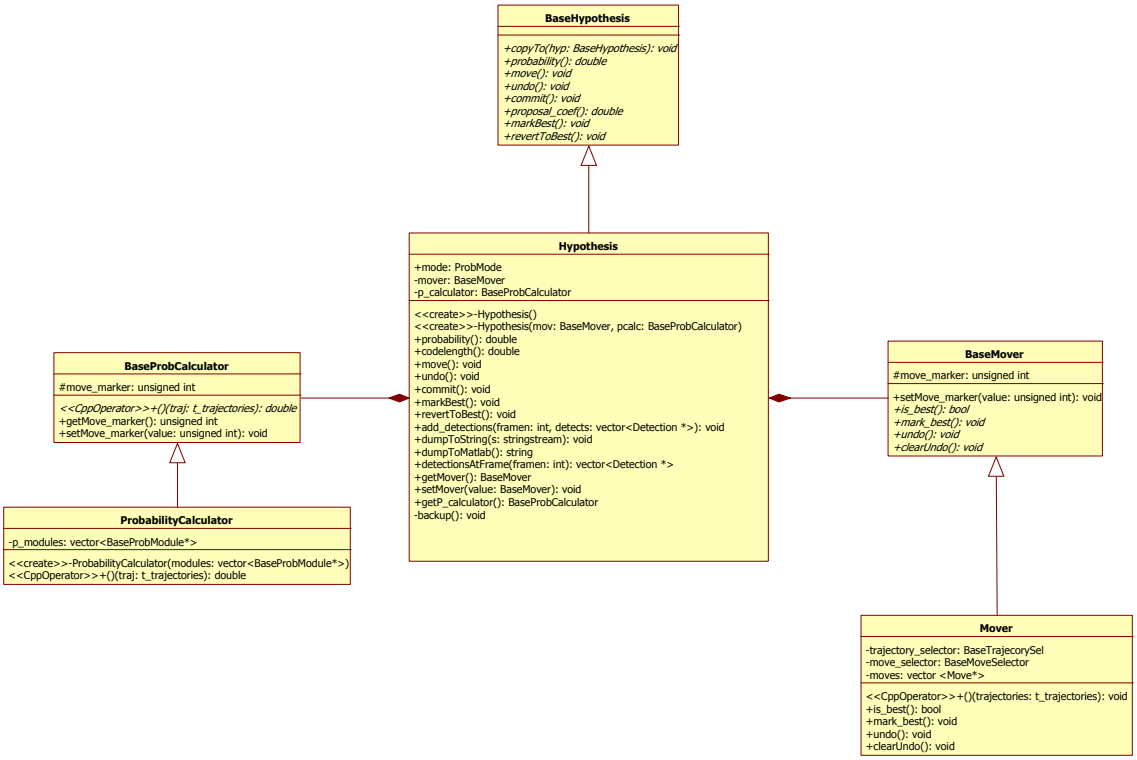


Figure 8.2: Main classes of the Monte Carlo Markov Chain Data Association unit

Output generation

After a track goes past MCMCDA window, it must be optimized and transformed into a suitable format. For this purpose, I decided to use the Kalman filter. As the initial kalman state prediction, I use position combined with velocity of the first detection. And as a measurements, I use the klt motion estimates in between detections. The resulting optimized track is then saved to xml file. In the future, this may be exchanged for a database or other form of data storage.

Chapter 9

Evaluation of the system

For the purpose of testing, I decided to use Town Center dataset. The system exhibited ability to track individual throughout environment. However, it still suffers from several shortcomings.

First of all, false positive on the side of detector showed to have large impact on the overall performance. Especially, cases when individuals are detected more then once with different bounding boxes. In situations like this, system correctly associates one bounding box of an individual and is unable to discard the other one. As a result, these detections tend to be paired with false positives, thus forming false tracks. Even incorporation of background-foreground ratio probability module has not led to solution of this problem, because involved true positive exhibits good foreground ratio.

The second most common problem are identity switches. In the case of AdaBoost module, these are cause by fact that humans tend to wear similar set of colors. Especially colors like white and black seem to be very common. From the side of motion probability module, these tend to happen in border regions of the frame.

In conclusion, system is not able to perform high accurancy individual tracking yet. Further work must be done to adress these issues. However, it is able to perform general crowd motion estimation. Thus, it may find applications in fields, such as customer behavior analysis, where information about general interest of people is more important than exact tracking of an individual.

Chapter 10

Conclusion

Proposed system was successfully implemented using C++ 11 and OpenCV computer vision library. Furthermore, it was tested on Town Center dataset provided by Benfold and Reid [12]. Evaluation has shown that it has ability to track individuals, however, it still suffers from problems related to false positives and identity switches. Therefore, it may be used in fields were exact tracking of each individual is not as important as estimation of behavior of the whole crowd.

In the future, accuracy of the system should be further improved either by revising current probability modules, or by creating new ones. Moreover, system may be extended to support trajectory storage in some form of database system. Another extension may be the implementation of a central server collecting tracks from multiple cameras with capability of performing analysis.

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2009.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[3] Takeo Kanade Carlo Tomasi. Detection and tracking of point features. Technical report, Carnegie Mellon University, 1991.

[4] Mike Stephens Chris Harris. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, 1988.

[5] Carlo Tomasi Jinabo Shi. Good features to track. *IEE Conference on Computer Vision and Pattern Recognition*, 1994.

[6] Reinhard Klette. *Concise Computer Vision: An Introduction into Theory and Algorithms (Undergraduate Topics in Computer Science)*. Springer, 2014.

[7] Colin Campbell Kristin P. Bennett. Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2000.

[8] Bill Triggs Navneet Dalal. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[9] Bernt Scheine Pietro Perona Piotr Dollár, Christian Wojek. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[10] Faith Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. *CVPR*, 2005.

[11] Nicolas Privault. *Understanding Markov Chains: Examples and Applications (Springer Undergraduate Mathematics Series)*. Springer, 2013.

[12] Ben Benfold Ian Reid. Stable multi-target tracking in real-time surevillance video. *CVPR*, 2011.

[13] Zdeněk Sojma. Object tracking in video. Master's thesis, Brno University of Technology, 2011.

[14] Shankar Sastry Songhwai Oh, Stuart Russel. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 2009.

[15] Shankar Sastry Songhwai Oh, Stuart Russel. Markov chain monte carlo data association for multi-target tracking. *IEEE Transactions on Automatic Control*, 2009.

[16] Marc Pollefeys Yakup Genc Sudipta N. Sinha, Jan-Michael Frahm. Gpu-based video feature tracking and matching, 2006.

[17] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[18] Xiaoou Tang Hueng-Yeung Shum Yichen Wei, Jian Sun. Interactive offline tracking for color objects. *ICCV*, 2007.

[19] Tobert E. Schapire Yoav Freund. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, 1996.