

# Multi Sensor Loitering Detection Using Online Viterbi

Håkan Ardo  
Centre for Mathematical Sciences  
Lund University, Sweden  
ardo@maths.lth.se

Kalle Åström  
Centre for Mathematical Sciences  
Lund University, Sweden  
kalle@maths.lth.se

## Abstract

*In this paper the problem of loitering detection in image sequences involving situations with multiple objects is studied. A multi camera approach is used to incorporate data from several cameras viewing the same scene. A Hidden Markov Model describing the movements of a varying number of objects as well as their entries and exits is used. The maximum likelihood over all possible state sequences is found using online Viterbi optimisation. Previously similar models have been used for single camera setups. In this paper the technique is extended to allow several cameras. The model is also made less sensitive to uninteresting objects occluding the region of interest, by integration over their effect on the observation probabilities. Finally the on-line Viterbi optimisation technique is extended to be able to produce its output after a constant number of frames. In previous work this delay varied with the complexity of the data. The resulting system is tested on the PETS2007 dataset scenarios S0-S2, with promising result.*

## 1. Introduction

Detecting simple events, such as a pedestrian entering a shop, a vehicle turning left at an intersection, or a person loitering, can be done by using a robust tracker. By placing restrictions on the tracker, such as “vehicles may only appear at the borders of the image and not in the middle of the intersection”, the event detection will be a simple matter of checking the endpoints of the tracks produced, and, in case of loitering, the time spend in the scene. In this paper we present a tracker that uses online Viterbi optimisation to find the set of tracks with maximum likelihood conforming to constraints as the one mentioned above.

### 1.1. Previous Work

In [1] an online system is presented that tracks a varying number of multiple moving objects. The entire state space is modelled by a Hidden Markov Model (HMM) [7], where each state represents a configuration of objects in the scene and the state transactions represent object movements as well as the events of objects entering or leaving the scene.

The solution is found by optimising the observation likelihood over different state sequences. Results are generated with several frames delay in order to incorporate information from both past and future frames in the optimisation. This delay varies depending on the input data.

In this paper that method is extended to (i) handle multiple cameras, (ii) produce results after a constant delay and (iii) improve the performance of objects moving outside the state space considered. The later extension improves the tracking results when there are objects outside some region of interest that occludes that region.

A very simple object model is used. It states that all objects are boxes of some given dimension and the observations made are the result of background foreground segmentation. The boxes are projected into the camera images and the probability of all pixels within the projected boxes being foreground and all pixel outside the boxes being background is calculated. By combining data from several cameras and using the efficiency of HMM to optimise over different state sequences, it is shown that this simple model can actually achieve reliable results.

For single target tracking the particle filter have been a successful solution. Extending this to handle a varying number of objects is not straightforward as the state now contains a discrete component, the number of objects currently present. One common way is to use one particle filter for each object, but that means that the problems of track initialisation, track termination and data association has to be solved separately, which is not the case when those events are modelled within the state space c.f. [4].

Most real-time HMM-based trackers [6], [5] and [3] do not use the standard Viterbi dynamic programming algorithm [7], which finds the global maximum likelihood state sequence. The main problem of using that algorithm is that it requires the entire set of future observations to be available. Instead they estimate the state posterior distribution given the past observations only. The particle filter also results in this kind of posterior state distribution, which means that both the particle filter and this kind of HMM trackers suffer from the problem of trying to estimate the single state of the system from this distribution. Later processing stages or data-displays usually requires a single state and not an

distribution.

Common ways to do this is to estimate the mean or the maximum (MAP) of this posterior distribution, but this have a few problems:

1. A mean of a multimodal distribution is some value between the modes. The maximum might be a mode that represents a possibility that is later rejected. In this paper we instead use optimisation that considers future observation and thereby chooses the correct mode.
2. In the multi object case the varying dimensionality of the states makes the mean value difficult to define. In [4] it is suggested to threshold the likelihood of each object in the configurations being present. Then the mean state of each object for which this likelihood is above some threshold is calculated separately.
3. Restrictions placed in the dynamic model, such as illegal state transactions, are not enforced, and the resulting state sequence might contain illegal state transactions. For the particle filter also restrictions in the prior state distribution might be violated. In [4] for example, the prior probability of two objects overlapping in the 3D scene is set to zero, as this is impossible. However the output mean state value may still be a state where two objects overlap, as the two objects may originate from different sets of particles. The problem is that only single states are considered. In the suggested approach state sequences are considered, which means that impossible state transactions will never appear in the results. Neither will impossible states, as the optimisation produces a single state sequence as the optimum, and there never is any need to calculate the mean over different states.

In [1] a novel modification to the Viterbi algorithm [7] were suggested. It allows it to handle infinite time sequences and still produce the global optimum. However, there is a delay of several frames between obtaining an observation and the production of the optimum state for that frame.

Another problem addressed in the same paper is that when considering multiple objects, the state space becomes huge. Typically at least some 10000 states is needed for a single object, and to be able to track  $N$  objects simultaneously that means  $10000^N$  states. An exact solution is no longer possible for real time applications. A possible approximation that can be used to compute results in real time were suggested: Only evaluate the  $M$  most likely states in each frame.

It was also shown that it is possible, to assess whether this approximation actually found the global optimum or not. This could be useful in an off line calibration of the approximation parameter  $M$ .

## 1.2. Paper overview

The paper is organised as follows. The theory behind hidden Markov models is described in Section 2. This Section begins with a review of the basic definition and the classic Viterbi optimisation. Then the extensions from [1] to handle infinite time sequences, Section 2.2, and infinite state spaces, Section 2.3 are briefly described. Section 3 describes how to use the HMM for multi target tracking, and includes our extension for using multiple cameras, Section 3.3, producing result after a constant delay, Section 3.4 and handle occluding objects not tracked, Section 3.5. Experimental verification is given in Section 4, and Section 5 concludes.

## 2. Hidden Markov models

A hidden Markov model, c.f. [7], is defined as a discrete time stochastic process with a set of states,  $S = S_0, \dots, S_N$  and a constant transitional probability distribution  $a_{i,j} = p(q_{t+1} = S_j | q_t = S_i)$ , where  $Q = (q_0, \dots, q_T)$  is a state sequence for the time  $t = 0, 1, \dots, T$ . The initial state distribution is denoted  $\pi = (\pi_0, \dots, \pi_N)$ , where  $\pi_i = p(q_0 = S_i)$ . The state of the process cannot be directly observed, instead some sequence of observation symbols,  $O = (O_0, \dots, O_T)$  are measured, and the observation probability distribution,  $b_j(O_t) = b_{j,t} = p(O_t | q_t = S_j)$ , depends on the current state. The Markov assumption gives that

$$p(q_{t+1} | q_t, q_{t-1}, \dots, q_0) = p(q_{t+1} | q_t) \quad (1)$$

and the probability of the observations satisfies

$$p(O_t | q_t, q_{t-1}, \dots, q_0) = p(O_t | q_t). \quad (2)$$

### 2.1. Viterbi optimisation

From a hidden Markov model  $\lambda = (a_{i,j}, b_j, \pi)$  and an observation sequence,  $O$ , the most likely state sequence,  $Q^* = \operatorname{argmax}_Q p(Q | \lambda, O) = \operatorname{argmax}_Q p(Q, O | \lambda)$ , to produce  $O$  can be determined using the classical Viterbi optimisation [7] by defining

$$\delta_t(i) = \max_{q_0, \dots, q_{t-1}} p(q_0, \dots, q_{t-1}, q_t = S_i, O_0, \dots, O_t). \quad (3)$$

For  $t = 0$ ,  $\delta_0(i)$  becomes  $p(q_0 = S_i, O_0)$ , which can be calculated as  $\delta_0(i) = \pi_i b_{i,0}$ , and for  $t > 0$  it follows that  $\delta_t(i) = \max_j (\delta_{t-1}(j) a_{j,i}) \cdot b_{i,t}$ . By also keeping track of  $\psi_t(i) = \operatorname{argmax}_j (\delta_{t-1}(j) a_{j,i})$  the optimal state sequence can be found by backtracking from  $q_T^* = \operatorname{argmax}_i \delta_T(i)$ , and letting  $q_t^* = \psi_{t+1}(q_{t+1}^*)$  for  $t < T$ .

### 2.2. Infinite time sequences

To handle the situations where  $T \rightarrow \infty$  consider any given time  $t_1 < T$ . The observation symbols  $O_t$ , for  $0 \leq t \leq t_1$ ,

have been measured, and  $\delta_t(i)$  as well as  $\psi_t(i)$  can be calculated. The optimal state for  $t = t_1$  is unknown. Consider instead some set of states,  $\Theta_t$ , at time  $t$  such that the global optimum  $q_t^* \in \Theta_t$ . For time  $t_1$  this is fulfilled by letting  $\Theta_{t_1} = S$ , the entire state space. For  $\Theta_t, t < t_1$ , shrinking sets of states can be found by letting  $\Theta_t$  be the image of  $\Theta_{t+1}$  under  $\psi_{t+1}$ , such that

$$\Theta_t = \{S_i | i = \psi_{t+1}(j) \text{ for some } S_j \in \Theta_{t+1}\}. \quad (4)$$

If the dependencies of the model is sufficiently localised in time, then for some time  $t_2 < t_1$ , there will be exactly one state  $q_{t_2}^*$  in  $\Theta_{t_2}$ , and the optimal state  $q_t^*$  for all  $t \leq t_2$  can be obtained by backtracking from  $q_{t_2}^*$ . No future observations made can alter the optimum state sequence for  $t \leq t_2$ .

### 2.3. Infinite state spaces

The problem with using the Viterbi optimisation for large state spaces is that  $\delta_t(i)$  has to be calculated and stored for all states  $i$  at each time  $t$ . By instead only storing the  $M$  largest  $\delta_t(i)$  and an upper bound,  $\delta_{\max}(t)$  on the rest, significantly less work is needed. If  $M$  is large enough, the entire optimal state-sequence might be found by backtracking among the stored states. It is also possible to verify if this is the case or not for a given example sequence and a given  $M$  by using  $\delta_{\max}(t)$ . If the global optimum were not found, then  $M$  could be increased and the algorithm executed again, or an approximate solution could be found among the stored states. Typically the algorithm is executed off-line for some example sequences to decide how large an  $M$  is needed, and then when running live this value is used and approximate solutions are found. The details of this algorithm and a proof of it's correctness is presented in [1].

## 3. Using HMM for tracking

### 3.1. Single object tracking

An HMM such as described above can be used for tracking objects in a video sequence produced by a stationary camera. Initially we assume that the world only contains one mobile object and that this object sometimes is visible in the video sequence and sometimes located outside the scene.

The state space of the HMM, denoted  $S^1$ , is constructed from a finite set of grid points  $X_i \in \mathbb{R}^2, j = 1, \dots, N$  typically spread in a homogeneous grid over the image. The state  $S_i$  represents that the mass centre of the object is at position  $X_i$  in the camera coordinate system. A special state  $S_0$ , representing the state when the object is not visible, is also needed.

The observation symbols of this model will be a binary background/foreground image,  $O_t : \mathbb{R}^2 \rightarrow \{0, 1\}$ , as produced by for example [2]. By analysing the result

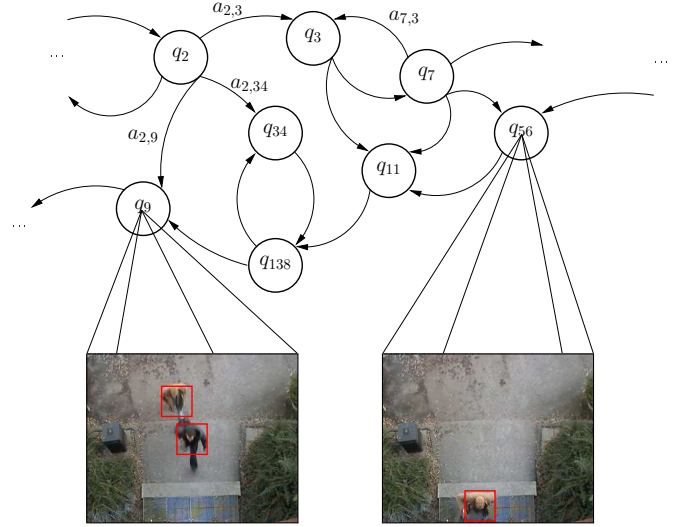


Figure 1: Illustration of the HMM used to track multiple objects. Each state represents a configurations of objects in the scene. The transactions represents object movements as well as the events of objects entering or leaving the scene.

of the background/foreground segmentation algorithm on a sequence with known background and foreground, the constant probabilities

$$p_{fg} = p(O_t(x) = 1 | x \text{ is a foreground pixel}) \quad (5)$$

and

$$p_{bg} = p(O_t(x) = 0 | x \text{ is a background pixel}) \quad (6)$$

can be calculated. Typically these are well above 1/2, and it is here assumed that they are constant over time and does not depend on  $x$ .

The shape of the object when located in state  $S_i$ , can be defined as the set of pixels,  $C_{S_i}$ , that the object covers when centred in at this position. This shape can be learnt from training data off line. As there is only one object in the world, when the HMM is in state  $S_i$ , the pixels in  $C_{S_i}$  are foreground pixels and all other pixels are background pixels. The probability,  $b_{i,t} = p(O_t | q_t = S_i)$ , of this is

$$b_{i,t} = \prod_{x \in C_{S_i}} [O_t(x)p_{fg} + (1 - O_t(x))(1 - p_{fg})] \cdot \prod_{x \notin C_{S_i}} [(1 - O_t(x))p_{bg} + (O_t(x))(1 - p_{bg})] \quad (7)$$

and thereby all parts of the HMM are defined.

### 3.2. Multi object HMMs

To generalise the one object model in the previous section into several objects is straightforward. For the two object

case the states become  $S_{i,j} \in S^2 = S \times S$  and the shapes,  $C_{S_{i,j}} = C_{S_i} \cup C_{S_j}$ . The transitional probabilities become  $a_{i_1 j_1 i_2 j_2} = a_{i_1 i_2} \cdot a_{j_1 j_2}$ . Figure 1 illustrates such a model.

Solving this model using the Viterbi algorithm above gives the tracks of all objects in the scene, and since there is only one observation in every frame, i.e. the background/foreground segmented image, no data association is needed. Also, the model states contain the entry and the exit events, so this solution also gives the optimal entry and exit points.

There is however one problem with this approach. The number of states increases exponentially with the number of objects and in practice an exact solution is only computationally feasible for a small number of objects within a small region of space.

### 3.3. Using multiple cameras

Extending this to multiple overlapping or non-overlapping cameras is relatively straightforward. By calibrating the set of cameras and identifying a common coordinate system for the ground plane, the objects centres,  $X_k$ , can be modelled as moving in this common coordinate system. Thereby a single HMM modelling the events on this ground plane can be used. The observations for this model is the images from all the cameras. Using the calibration of the cameras, each centre point can be projected into the shape of the object in each of camera images  $C_{X_k}^c$  where  $c = 1, 2, \dots$ , represents the different cameras.  $C_{X_k}^c$  might be the empty set if an object at position  $X_k$  is not visible in camera  $c$ . By indexing Equation 7 on the camera  $c$ , with  $O_t^c$  the background/foreground image produced from camera  $c$ ,

$$b_{i,t}^c = \prod_{x \in C_{S_i}^c} [O_t^c(x)p_{fg} + (1 - O_t^c(x))(1 - p_{fg})] \cdot \prod_{x \notin C_{S_i}^c} [(1 - O_t^c(x))p_{bg} + (O_t^c(x))(1 - p_{bg})], \quad (8)$$

the total observation probability becomes

$$b_{i,t} = \prod_c b_{i,t}^c. \quad (9)$$

### 3.4. Using constant delay

For real time application it is interesting to restrict the maximum delay,  $T_{\max}$ , between receiving a frame and outputting the result for that frame. That would also make the processing time of each frame and the memory needed more constant. This can be achieved by summing up the total likelihood converging in each node while backtracking,

$$\Delta_t(i) = \sum_{\psi_{t+1}(j)=i} \Delta_{t+1}(j). \quad (10)$$

For the last frame  $t_1$ , we set  $\Delta_{t_1}(i) = \delta_{t_1}(i)$ . As the algorithm is presented above, the backtracking should continue until  $\Delta_{t_2}(i) = 0$  for all  $i$  but one. But if instead the backtracking is terminated while  $\Delta_{t_2}(i) > 0$  for several  $i$ , the resulting state for  $t_2$  could be chosen as  $q_{t_2} = \operatorname{argmax}_i \Delta_{t_2}$ .

By doing so a decision has been made for  $t_2$ . To ensure that the resulting state sequence is a legal one, this decision has to be propagated forward. For approximate solutions this could be done by letting  $\delta_t(i) = 0$  for all  $i$  and  $t$  that backtracks to any other state than  $q_{t_2}$ . For the exact solutions it is enough to let  $\psi_{t_2}(i) = -1$  for all  $\{i | \Delta_{t_2}(i) > 0, i \neq q_{t_2}\}$ .

The additions introduced by evaluating (10) replace conditionals in the original backtracking algorithm. So, if conditionals and additions have the same cost, this part will have the same cost as the original backtracking. The extra computational effort needed is to propagating the decisions made forward. It works in much the same way as the backtracking and have the same complexity. For a given length this means that the constant delay improvement doubles the amount of work needed as compared to the standard backtracking. But the idea is to make the length lower, which also means that memory is saved. For the overall system the time spent backtracking is negligible, as almost all time is spent in the forward propagation of the Viterbi optimisation, Equation 3.

### 3.5. Using Region of Interest

In many cases it is neither interesting nor computational possible to consider all objects visible in the camera field of view. Typically the state space is restricted to only consider objects within a specific region of interest. But this leads to problems along the border of this region as objects moving outside this region, but close enough to occlude it in the camera view, will affect the background/foreground segmentation within. Regardless of how the region of interest is chosen (including the entire image), there will always be objects outside this region but close enough to have some part of them project into the region in the image.

In [1] it was suggested to solve this by using a post-processing step that discards objects located at the very border of the region of interest. But if there is a lot of activity outside the region this means that a lot of uninteresting objects has to be modelled, which increases the computational time. It also increases the number of possibilities, which means that  $M$  has to be chosen larger. That increases the computational time even further. Also, the true state of those objects outside is not in the state space, which means that their state will be projected into the state space. That might generate strange results.

We suggest to instead model the entire world as a single ground plane with an infinite number of equally shaped ob-

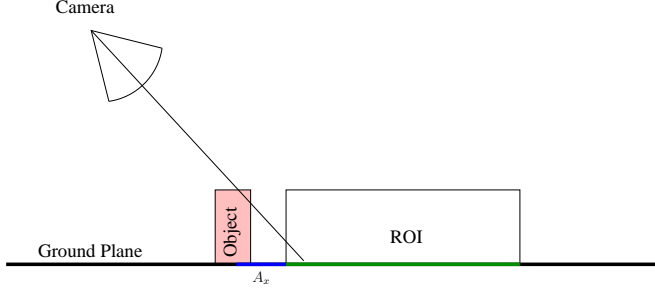


Figure 2: This setup shows a camera viewing a ground plane at a tilted angle. A region of interest is specified in 3 dimensions with its height equal to the height of the objects. The thick blue line indicates the region in which objects outside the region of interest will occlude the region of interest.

jects moving along this plane. The positions of the objects outside a given region of interest are considered uninteresting and are integrated out. This means that  $S_i$  will represent the state “one object within the region of interest at position  $i$ , and any number of objects at any positions outside”.

To form an observation-model for this kind of states some prior distribution of how many objects are located within a certain area has to be assumed. It could for example be assumed that the number of objects present within an area,  $A$ , is Poisson distributed with mean  $\lambda A$ , where  $\lambda$  is a parameters specifying how many objects are expected per  $m^2$ ,

$$p_o(n, A) = p(n \text{ objects present in } A) = \frac{(\lambda A)^n e^{-\lambda A}}{n!}. \quad (11)$$

Using the camera calibration, the geometry of the setup such as shown in Figure 2 can be calculated. The region of interest is a three dimensional box, with the same height as all the objects. For every pixel  $x$  the area  $A_x$  of the region outside the region of interest where objects occlude the region of interest can be calculated. The probability of the background at that pixel being visible, given that no object within the region of interest occludes it is  $p_o(0, A_x) = e^{-\lambda A_x}$ . The probability of the background at that pixel being occluded given that an object within the region of interest occludes it is of course 1. This can be introduced into the observation probability in (7), by replacing  $p_{bg}$  with

$$p'_{bg}(x) = e^{-\lambda A_x} p_{bg} + (1 - e^{-\lambda A_x}) (1 - p_{fg}), \quad (12)$$

which is no longer constant, but varies over the pixels,  $x$ .

The prior model assumed should then also be used in the state probability by adding a term,  $p_o(n, A)$  to (9), where

$A$  is the area on the ground plane covered by the region of interest, and  $n$  is the number of objects present within. With this modification (9) becomes

$$b_{i,t} = \frac{(\lambda A)^n e^{-\lambda A}}{n!} \prod_c b_{i,t}^c. \quad (13)$$

All the calculations can be performed offline and stored in an calibration image that for each pixel,  $x$ , stores the value of  $p'_{bg}(x)$ . The extra work needed online is then reduced to a pixel lookup in this image. This is the same amount of work needed when a binary mask is used to mask out regions permanently occluded. To use this kind of soft mask instead of a binary one makes it possible to use data from cameras mounted at an lower angle. A binary mask removing all pixels that might originate from outside the region of interest in the PETS dataset (see Section 4 and Figure 3) would render the entire image from camera 4 masked out, and only very small portions of the images from camera 1 and 2 unmasked.

## 4. Experiments

Tests were performed on the PETS 2007 dataset<sup>1</sup>. It consists of 8 different scenarios, each recorded with four cameras from four different angles, Figure 3. In addition to this there is a background sequence that can be used for calibrating the system. The first three scenarios, S0, S1 and S2 where analysed to automatically detect loitering. Loitering is defined as a person who enters the field of view of camera 3, and remains within the scene for more than 60 seconds. The dataset comes with calibration parameters for the cameras, that were calculated from markers on the floor using the Tsai camera model [8]. There were some problems with the calibration initially released for camera 3. It was recalibrated by manually identifying the markers in the image. The camera parameters were the estimated using the freely available Tsai Camera Calibration Software<sup>2</sup> by Reg Willson. Since then a new version of the camera 3 calibration has been released, but the tests presented here have not been reevaluated using this calibration data.

The ground plane was discretised into a uniform 74x42 grid reaching from  $(x, y) = (-4, -2)$  to  $(x, y) = (4, 2.5)$  in world coordinates (meters) as defined by the camera calibration’s provided. This roughly corresponds to the field of view of camera 3. The objects were modelled as 0.44x0.44x1.8 m boxes. Those boxes were projected into each of the four camera images, and bounding boxes parallel to the image axes where fitted. This allows the observation probability of each state to be calculated very fast using integral images.

<sup>1</sup><http://pets2007.net/>

<sup>2</sup><http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html>

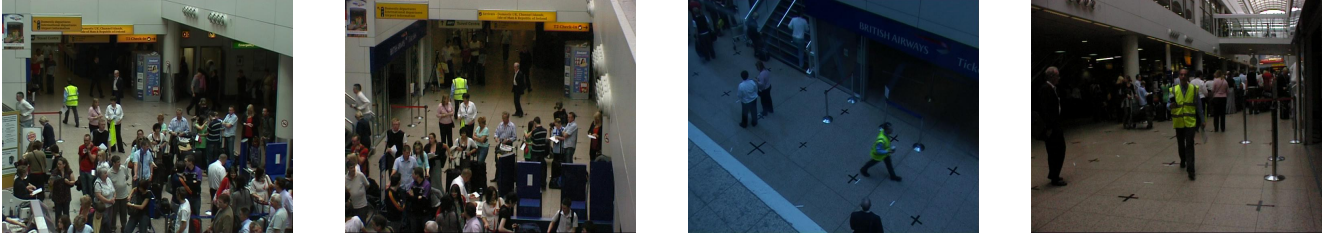


Figure 3: Example frames from the PETS 2007 dataset. From left to right, camera 1 to 4.

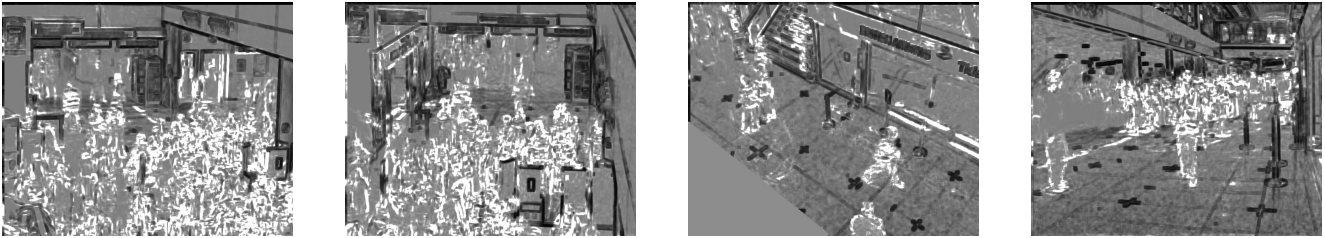


Figure 4: Example result from the background/foreground segmentation algorithm used as input. The images corresponds to the frames in Figure 3

Objects detected on the very border of the grid were ignored since such detections often correspond to object movements outside the grid. The number of frames an object stayed strictly in the interior of the grid were counted, and if this exceeded 1500 (60 s), the loiterer alarm were raised. The sequence  $S_0$  contains no loiters, while  $S_1$  and  $S_2$  contains one each.

#### 4.1. Background/Foreground segmentation

As input to the algorithm a correlation based background/foreground segmentation, where used. It compares  $8 \times 8$  blocks of the background model with the corresponding block in current frame using the correlation coefficient. For each pixel the probability of it being foreground is calculated using a probabilistic background/foreground segmentation. A typical result is shown in Figure 4. For camera 3 a mask where generated manually masking the pixels in it's lower left where some static object is located between the camera and the scene. The mask is used to set the foreground probability of all occluded pixels to 0.5. To make sure that the estimate of the background model had converged before the tracking started, the background sequences were concatenated to each of the scenarios. The tracking was not started until the real sequence begun. There is no initialisation of the state for the first frame. The scene is assumed to be empty. If this is not the case the first seconds of the tracking result contains errors. But after that the tracker typically converges to a correct state.

| $M$  | Cameras | Runtime/frame (ms) |
|------|---------|--------------------|
| 10   | 1       | 1                  |
| 10   | 2       | 2                  |
| 10   | 3       | 2                  |
| 10   | 4       | 4                  |
| 100  | 1       | 15                 |
| 100  | 2       | 21                 |
| 100  | 3       | 30                 |
| 100  | 4       | 42                 |
| 1000 | 1       | 200                |
| 1000 | 2       | 281                |
| 1000 | 3       | 429                |
| 1000 | 4       | 526                |

Table 1: Mean frame processing time for different values of the  $M$  parameter described in Section 2.3 using different number of cameras.

#### 4.2. Result

To test the real time performance of the HMM tracker, it's runtime where measured on a 2.40GHz P4, for different values on  $M$  and different number of cameras. The background/foreground segmentation was precalculated, so it's runtime is not considered. For  $M = 1$  the algorithm becomes equivalent to a greedy algorithm changing to the most likely state in each frame. The result is shown in Table 1.

The performance of the system was measured by eval-

| $M$ | Cameras used | Loiter detections |    |    | Tracker OK (%) |     |
|-----|--------------|-------------------|----|----|----------------|-----|
|     |              | S0                | S1 | S2 | Frames         | Ids |
| 100 | 3            | 0                 | 1  | 1  | 84             | 73  |
| 100 | 1 3          | 0                 | 1  | 1  | 76             | 62  |
| 100 | 2 3          | 0                 | 1  | 2  | 85             | 80  |
| 100 | 3 4          | 0                 | 1  | 1  | 84             | 73  |
| 100 | 1 2 3        | 0                 | 1  | 2  | 88             | 84  |
| 100 | 1 3 4        | 0                 | 1  | 1  | 76             | 62  |
| 100 | 2 3 4        | 0                 | 1  | 2  | 85             | 81  |
| 100 | 1 2 3 4      | 0                 | 1  | 1  | 88             | 84  |

Table 2: Tracking result using different sets of cameras. The number of loitering detections found in each of the three sequences are shown. S0 contains no loiters, while S1 and S2 contains one each. The “Tracker OK Frames” column shows the percentage of frames the the algorithm has detected objects less than 1m from the objects in the ground truth. “Tracker OK Ids” shows the percentages of object frames where the id-number is not mixed up between objects.

uation how successful the loitering detection is in different setups. Each setup uses a different set of cameras. The correct result is to find one loiter in each of S1 and S2 and zero in S0. In three of the setups there were an extra loiter detected, which actually is the same loiter detected twice due to an id-mixup between two objects.

The dataset used comes with ground truth data. This data were compared to tracks generated by the suggested algorithm. For each frame with ground truth, the tracking results were search for objects within 1m of the ground truth objects. If such objects could be found for all ground truth objects, that frame was considered OK. The percentage of good frames from S1 (ground truth for S0 and S2 were not available at the time) is presented In the “Tracker OK Frames” column of Table 2.

To also verify the id-numbers each track of the ground truth were mapped to the track it fitted closest among the results generated from the tracker. The total number of frames for each track in the ground truth where the ground truth is less than 1m from the tracked objects were counted. The percentage of such frames is presented in the “Tracker OK Ids” column.

The two “Tracker OK” columns should be interpreted as follows. If “Frames” is close to 100%, but “Ids” is lower, the objects were tracked, but the identities mixed up. If “Frames” is low some object were missed altogether.

## 5. Summary and Conclusions

In [1] a multi HMM model where proposed, to model multiple targets in one single model with the advantage of solving

all the problems of a multi target tracker by a Viterbi optimisation. This includes track initiation and termination as well as the model updating and data association problems. In this paper this model is extended to handle multiple cameras surveying the same scene. Also, the online Viterbi algorithm, suggested in the same paper, is extended making it possible to specify a maximum delay between the reception of a frame and the production of the tracking result for this frame. The produced tracks are still guaranteed to follow any restrictions placed on the model, such as “objects may not overlap” or “objects many only (dis)appear along the borders”. Finally this paper introduces a statistical model that makes it possible for the tracker to ignore uninteresting objects moving outside some region of interest by integration out their effect on the observation model.

The proposed algorithm is tested by building a loitering detection system and evaluating it on the scenario S0, S1 and S2 of the PETS 2007 dataset, with promising result.

To improve the result it should be fairly straightforward to plug in some more sophisticated object model, instead of the background/foreground segmentation, e.g. a pedestrian detector.

## References

- [1] H. Ardö, R. Berthilsson, and K. Åström. Real time viterbi optimization of hidden markov models for multi target tracking. In *IEEE Workshop on Motion and Video Computing*, 2007.
- [2] Håkan Ardö and Rikard Berthilsson. Adaptive background estimation using intensity independent features. *Proc. British Machine Vision Conference*, 03:1069–1078, 2006.
- [3] Mei Han, Wei Xu, Hai Tao, and Yihong Gong. An algorithm for multiple object trajectory tracking. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, volume 01, pages 864–871, Los Alamitos, CA, USA, 2004. IEEE Computer Society.
- [4] Michael Isard and John MacCormick. Bramble: A bayesian multiple-blob tracker. In *Proc. 8th Int. Conf. on Computer Vision, Vancouver, Canada*, pages 34–41, 2001.
- [5] Jien Kato, T. Watanabe, S. Joga, Ying Liu, and H. Hase. An hmm/mrf-based stochastic framework for robust vehicle tracking. *IEEE Transactions on Intelligent Transportation Systems*, 5(3):142–154, 2004.
- [6] Javier Movellan, John Hershey, and Josh Susskind. Real-time video tracking using convolution hmms. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, 2004.
- [7] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, 1989.
- [8] R. Tsai. An efficient an accurate camera calibration technique for 3D machine. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 364–374, 1986.



Figure 5: Resulting output of the tracker from frames 700-900 of scenario S1. The red boxes indicates the detected and tracked objects in the four cameras. The green rectangle indicates the area with which objects are tracked.