Detecting pedestrians intending to enter a crosswalk using a HMM tracker and a novel predictor

Emanuel Hasselberg

Master's thesis
2013:E22

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Abstract

There is a demand for a more fluent and more efficient traffic. This can be achieved through more intelligent traffic light control. This thesis presents theory and an application in which people are tracked and their intentions to cross a crosswalk is predicted with a novel prediction algorithm based on Markov theory. The background segmentation and tracking algorithms was based on already known cross-correlation and HMM-methods.

Based on the relatively small amount of training data the result for the novel predictor detecting persons "entering the crosswalk" for two different setups, a straight and a four way crossing, is 70% and 55% true positives with 5% and 2% false positives. For detection of someone that is "not entering the crosswalk" when there is a person in the area is 90% and 85% true positives with 15% and 25% false positive.

The results achived is good enough as a proof of concept that the theories are worth investigating further for these kind of applications. However, a lot of work would still be required before this is robust enough to be in a real traffic application.

Contents

1 Introduction

In these times, there is a desire for faster and more efficient traffic lights which could generate a more fluent flow in traffic for cars and pedestrians. This thesis will present a method for following pedestrians approaching a crosswalk and predict their intention to cross or not. An application has been developed and will be presented in the end, that could be used to make the traffic lights more efficient.

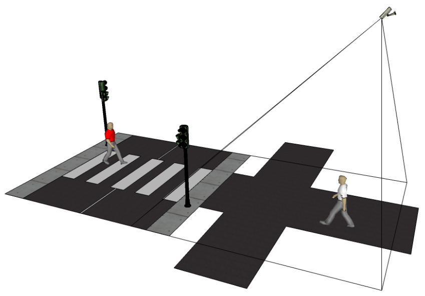


Figure 1: In the illustration there is a crosswalk and paths leading to it. A camera is placed so it can observe the paths and the entrance to the crosswalk. Here we would like to know: What is the probability that the person will enter the crosswalk.

1.1 Related work

There are many areas where pedestrian tracking is used as a component in a more complex system. Such as, tracking system for counting people in dense crowds [5], bicycle theft detection system [6], player tracking in sports videos for the analysis of player movement [7] and system for measuring the gaze directions of pedestrians in surveillance video [8]. A closely related publication is "Trajectory modeling for the identification of unusual incidents" [9], where the interest in unusual behavior is in focus. This application uses an object tracker [10] based on Active Shape Models. The trajectory's are predicted using a neural network and uses the history of the whole trajectory to match one of the learnt. This method could probably be adapted to work for this application as well but would require a lot of training data to make it statistically correct.

2 Problem Description

Todays implementations of traffic lights use as much information as possible, underground magnetic sensors for cars and radar for bicycles. The radar can detect fast pacing pedestrians as well, but not as good as desired. This information is then processed to make the traffic lights as optimized as possible, making traffic fluent and efficient.

The goal is to construct a sensor that gives good information to the traffic lights. The perfect predictor would give the probability that a pedestrian would enter the crossing at time t. That sensor would not only be an on off switch i.e. a person is coming we don't know for sure or even when. But a probability distribution over time t would be more informative i.e. a person is coming with a probability of 80% in 5 s.

2.1 Approach and Limitations

In this thesis the approach will be to track the pedestrians using a camera overveiwing a crosswalk from above. By continiously learning the paths of the pedestrians observed, the predictor can be built and updated online. No temperal information will be used in the algorithms, they are only based on spatial data.

The steps throughout the thesis is:
1. Background foreground segmentation, Section 3
2. Tracking the pedestrians using Hidden Markov Models, Section 4
3. Calculate the probability for entering the crosswalk, Section 5
4. Update the predictor, Section 5.1

3 Background Foreground Segmentation

Background/Foreground (bg/fg) segmentation is used to segment out moving objects from a sequence of images. For example in a video sequence of an empty street where persons are crossing. The bg/fg segmentation should tag the pixels the persons are in as fg and the rest as background. The segmentation algorithm used [1] in this thesis is based on cross correlation features. This section will only present a brief overview of this method.

When working with intelligent video it is often desirable to have a fast and reliable foreground/background segmentation to let the CPU work with other algorithms. First lets define what we want in a foreground/background segmentation. We want a probability image, P, such that $0 \leq P(x, y) \leq 1$ for all $x, y \in I$ where I is a set of all pixel coordinates in the image. Where $P(x_k, y_k)$ is the probability that the pixel at (x_k, y_k) is foreground. If P can be calculated for every frame, the foundation for the tracking algorithm is built.

This is to be achieved under the condition that the intensity in the picture can change very fast, not always global, but locally by clouds covering just some parts. This is made by comparing patches of 8x8 or 4x4. Call the background patch p , this is updated to represent the median background patch. Not only the median value is estimated, also the noise distribution in the patch. This can vary from patch to patch due to things moving in the background e.g. leafs moving in the wind. The foreground probability p_{fg} is given by comparing the current patch to the model with cross correlation. Figure 10c on page 13 shows an example.

4 Hidden markov models

In this section we present the method used to track pedestrians, based on a Hidden Markov Model (HMM). The idea behind HMM is to find the best track over a sequence of observations O not only the last observation O_τ. A downside is that the result is delayed with several frames.

The HMM is defined as a discrete stochastic process with a set of n states, $\mathbb{S} = \{S_0, ..., S_n\}$. The states will be explained in the next section, for now

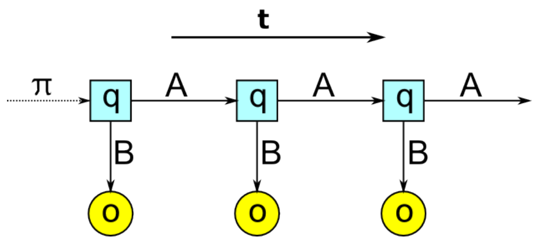


Figure 2: The state q is not directly observable. Instead the observation o is observed.

see it as the position where the person is standing. And a transitional probability distribution, $a_{i,j}$, the probability a person moves from state i to state j in one frame, $a_{i,j} = p(q_{t+1} = S_j | q_t = S_i)$. The state sequence is denoted $\mathcal{Q}_{0...\tau} = (q_0, ..., q_\tau)$ for the time $t = 0, ..., \tau$, i.e. the positions of the person at the time t. The initial state distribution i.e. the probability that a person is standing in a specific state at $t = 0$ is $\pi = (\pi_0, ..., \pi_n)$, where $\pi_i = p(q_0 = S_i)$. The sequence $\mathcal{Q}_{0...\tau}$ is the true trajectory of the person, this cannot be directly observed. Instead a sequence of observations is made through the camera $\mathcal{O}_{0...\tau} = (\mathbf{o}_0, ..., \mathbf{o}_\tau)$. The observation probability distribution $b_j(\mathbf{o}_t) = b_{j,t} = p(\mathbf{o}_t | q_t = S_j)$, is a distribution that gives the probability that the observation $\mathbf{o}_t$ is made when the pedestrian has state S_j, more of this in section 4.2. The markov assumption gives that the next state is only dependent of the current state $p(q_{t+1} | q_t, q_{t-1}, ..., q_0) = p(q_{t+1} | q_t)$, and the same for observations $p(\mathbf{o}_t | q_t, q_{t-1}, ..., q_0) = p(\mathbf{o}_t | q_t)$, i.e. the information the camera gives about the position the person standing in, is depending only on the current frame.

4.1 The States a.k.a. The Grid

The HMM work with a set of states, every state is a possible configuration in the scene. For tracking, a space-grid is used the represent the states. In the case of tracking a single person the states are denoted as $\mathbb{S}^1$, constructed from a set the grid points $\mathbf{x}_i \in \mathbf{R}^2, i = 1, ..., n$. The grid is spread homogeneous over the area of interest (AOI). The state S_i is the state where a person has it's center at point $\mathbf{x}_i$. A special state is S_0 which is analogous to no persons at all in AOI. For the case when several persons are in the AOI the states in $\mathbb{S}^1$ are inadequate, a bigger state space is required. The expansion $S_{i,j} \in \mathbb{S}^2 = \mathbb{S}^1 \times \mathbb{S}^1$ is capable to handle two persons, a bigger expansion is straightforward. Figure 10b on page 13 shows an example of a grid and a state where two persons are present.

4.2 The observation probability $b_{j,t}$

In section 4 the observation probability distribution is declared as $b_j(\mathbf{o}_t) = b_{j,t} = p(\mathbf{o}_t | q_t = S_j)$. In this section we are going to define a way to take a foreground/background segmentation image denoted $O_t(\cdot)$ and calculate $b_{j,t}$. The observation $\mathbf{o}_t$ is $O_t(\cdot)$ where the dot stands for all pixels in the image.

The general idea to calculate $b_{j,t}$ is to generate a expected observation image $\hat{O}_t$ for state S_j there the set of pixels expected to be foreground are defined C_{S_j}. If this matches good against the observation O_t, a high probability for $b_{j,t}$ should apply. To construct C_{S_j} for a person its hard to get a object similar to a person, instead a box wide enough to hold a person is used. For state S_j a box is placed in the ground plane with position x_j and rotated to face the camera. This is then projected into the image creating the set C_{S_j}. For a state with

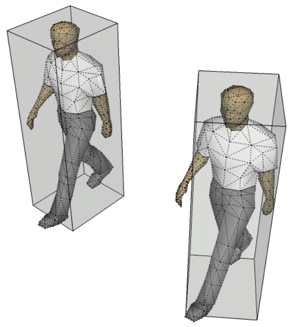


Figure 3: To the left a box covering the person. There are big areas that are outside the body of the person. By rotating the box towards the camera, as in the box to the rigth, there are less area outside the body.

multiple persons e.g. $S_{j,k}$, the set $C_{S_j} \cup C_{S_k}$ is used, the same as projecting two boxes into the image plane. Consider each pixel $o_t = O_t(x, y)$, separately. The generated expected background/foreground segmentation image $\hat{O}_t$, where $\hat{o}_t \in \{0, 1\}$ will differ from the true background/foreground segmentation image $\hat{O}'_t$, where $\hat{o}' \in \{0, 1\}$ due to for example,

- using a box as model for a person
- the state space is discrete so the positions don't match exactly with the true
- background/foreground segmentation is not perfect.

Now look at the probability $p(o_t|\hat{o}_t)$ i.e. that a pixel in the observation concur with the expected background/foreground segmentation pixel. Given $P(A|B) = \sum_i P(A|C_i)P(C_i|B)$ the probability can be re-written as

$$p(o_t|\hat{o}_t) = \sum_{\hat{o}'_t \in \{0,1\}} p(o_t|\hat{o}'_t)p(\hat{o'_t}|\hat{o}_t) \tag{1}$$

now we can look at the two parts, $p(o_t|\hat{o}'_t)$ and $p(\hat{o'_t}|\hat{o}_t)$.

$p(o_t|\hat{o}'_t)$ is the probability that the pixel is "correct", given that we know if o_t is in the true background/foreground segmentation $\hat{o}'_t$. By correct we mean if $\hat{o}'_t = 1$ (the pixel should be foreground) the probability is given by the foreground/background segmentation $o_t = p_{\text{fg}|c,\bar{l}}(c_x)$ for that pixel. This gives

$$\begin{aligned} p(o_t|\hat{o}'_t) &= \begin{cases} o_t & \text{if } \hat{o}'_t = 1 \\ 1 - o_t & \text{if } \hat{o}'_t = 0 \end{cases} \\ &= o_t\hat{o}'_t + (1 - o_t)(1 - \hat{o}'_t) \end{aligned}$$

which solves the first part in (1).

Next is the probability, $p(\hat{o'_t}|\hat{o}_t)$ the probability that the true observation is in the expected observation. It has four binary cases. These has to be estimated using observations with known state q_t and are denoted variable names p_{fg} and p_{bg}.

$$\begin{aligned}
p(\hat{o}'_t = 1|\hat{o}_t = 1) &= p_{\text{fg}} \\
p(\hat{o}'_t = 0|\hat{o}_t = 1) &= 1 - p_{\text{fg}} \\
p(\hat{o}'_t = 1|\hat{o}_t = 0) &= 1 - p_{\text{bg}} \\
p(\hat{o}'_t = 0|\hat{o}_t = 0) &= p_{\text{bg}}
\end{aligned}$$

These variables are typically well over 1/2 and are assumed constant and independent of time and position.

Now we can rewrite (1) as

$$\sum_{\hat{o}'_t \in \{0,1\}} p(o_t|\hat{o}'_t)p(\hat{o'_t}|\hat{o}_t) = \begin{cases} o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}}) & \text{if } \hat{o}_t = 1 \\ o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}} & \text{if } \hat{o}_t = 0 \end{cases}$$

To combine the evidence from all pixels observed into a single likelihood, they are considered independent and combined as a product.

$$b_{i,t} = \prod_{o_t \in \mathcal{C}_{S_i}} (o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}})) \cdot \prod_{o_t \notin \mathcal{C}_{S_i}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}}) \quad (2)$$

4.3 Rescaling

To calculate $b_{i,t}$ in (2) it's needed to iterate over all pixels in the image for all different sets $\mathcal{S}$. Instead by letting

$$\tilde{b}_{i,t} = \frac{b_{i,t}}{b_{0,t}} = \frac{\prod_{o_t \in \mathcal{C}_{S_i}} (o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}})) \cdot \prod_{o_t \notin \mathcal{C}_{S_i}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}})}{\prod_{o_t \in \mathcal{C}_{S_0}} (o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}})) \cdot \prod_{o_t \notin \mathcal{C}_{S_0}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}})}.$$

Where $\prod_{o_t \in \mathcal{C}_{S_0}} (o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}}))$ is by definition 1.
We can rewrite

$$\begin{aligned}
&\textstyle\prod_{o_t \notin \mathcal{C}_{S_0}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}}) = \\
&\textstyle\prod_{o_t \in \mathcal{C}_{S_i}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}}) \cdot \prod_{o_t \notin \mathcal{C}_{S_i}} (o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}})
\end{aligned}$$

and

$$\tilde{b}_{i,t} = \frac{b_{i,t}}{b_{0,t}} = \prod_{o_t \in \mathcal{C}_{S_i}} \frac{(o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}}))}{(o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}})}.$$

Finally we get a way we only have to take the product over the foreground pixels.

4.4 Using an integral image

Using the logarithm on $\tilde{b}_{i,t}$ does not change the optimal sequence , but replaces the product with a sum.

$$L_{i,t} = log\ \tilde{b}_{i,t} = log\ \frac{b_{i,t}}{b_{0,t}} = \sum_{o_t \in \mathcal{C}_{S_i}} log\ \frac{(o_t p_{\text{fg}} + (1-o_t)(1-p_{\text{fg}}))}{(o_t(1-p_{\text{bg}}) + (1-o_t)p_{\text{bg}})}$$

Making it possible to use an integral image. This will speed up calculations.

The integral image is a intermediate representation that in each pixel x, y contains the sum of the pixels above and to the left of x, y. That is, $I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$ where $I(x, y)$ is the integral image and $i(x, y)$ is the original image.

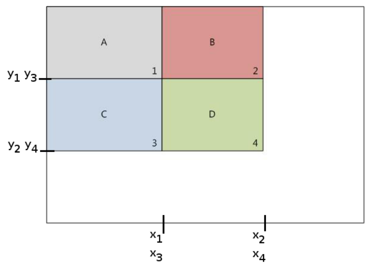


Figure 4: The sum in area A is $I(x_1, y_1)$. To calculate the sum in area D, its equal to $I(x_4, y_4) + I(x_1, y_1) - I(x_2, y_2) - I(x_3, y_3)$

4.5 Infinite Viterbi

Every frame has now a observation probability $b_{i,t}$ but we dont know what path the pedestrian is walking. Here is where infinite Viterbi comes in.

For an observation sequence $\mathcal{O}_{0\cdots\tau}$ and the HMM $\lambda = (a_{i,j}, b_j, \pi)$ the algorithm will try to find the most likely state sequence

$$\mathcal{Q}^*_{0\cdots\tau} = \underset{\mathcal{Q}_{0\cdots\tau}}{\operatorname{argmax}}\, p(\mathcal{Q}_{0\cdots\tau}|\lambda, \mathcal{O}_{0\cdots\tau})$$

that generated the observations $\mathcal{O}_{0\cdots\tau}$.

Infinite Viterbi is an algorithm that can solves this. The difference to classical viterbi [3] where a large state spaces is hard to handle, due to the masses of internal probability states needed to be stored, even if they are very small. Infinite viterbi only stores the m larges probability states and a upper bound on the rest which increases performance. The variable m should be manually decided by testing or calibration.

The idea behind the algorithm is that $p(\mathcal{Q}_{0\cdots\tau}|\lambda, \mathcal{O}_{0\cdots\tau})$ can be written as $\frac{p(\mathcal{Q}_{0\cdots\tau}, \mathcal{O}_{0\cdots\tau}|\lambda)}{p(\mathcal{O}_{0\cdots\tau})}$. Because $p(\mathcal{O}_{0\cdots\tau})$ does not depend on the state sequence $\mathcal{Q}^*_{0\cdots\tau}$, we can write $\mathcal{Q}^*_{0\cdots\tau} = \underset{\mathcal{Q}_{0\cdots\tau}}{\operatorname{argmax}}\, p(\mathcal{Q}_{0\cdots\tau}, \mathcal{O}_{0\cdots\tau}|\lambda)$. To find the optimum of this expression, we define $\delta_t(i) = \max_{q_o,\ldots,q_{t-1}} p(q_0, ..., q_{t-1}, q_t = S_i, \mathbf{o}_0, ..., \mathbf{o}_t)$. The path that gives the highest probability to be in state i at time t. Markov assumption gives that

$$\begin{aligned} p(q_0, ..., &q_{t-1} = S_j, q_t = S_i, \mathbf{o}_0, ..., \mathbf{o}_t) = \\ &p(\mathbf{o}_t|q_t = S_i)p(q_t = S_i|q_{t-1} = S_j)p(q_0, ..., q_{t-1} = S_j, \mathbf{o}_0, ..., \mathbf{o}_{t-1}) = \\ &b_{j,t}a_{j,i}p(q_0, ..., q_{t-1} = S_j, \mathbf{o}_0, ..., \mathbf{o}_t) \end{aligned}$$

and implies that $\delta_t(i)$ could be calculated iteratively.

At the same time, keep track of each $\delta_t(i)$ which $\delta_{t-1}(j)$ it came from as $\psi(i) = \underset{j}{\operatorname{argmax}}(\delta_{t-1}(j)a_{j,i})$. The optimal state sequence can be found by backtracking from $q^*_\tau = \underset{i}{\operatorname{argmax}}\, \delta_\tau(i)$, the best state at time τ. For $t < \tau$, a backtrack gives $q^*_t = \psi_{t+1}(q^*_{t+1})$.

4.6 Online Viterbi Optimization

Infinite Viterbi only handles situations where the sequence is of fixed time $0 < t < t_1$ and not when When $\delta_t(i)$ and $\psi_t(i)$ has been calculated for the observations to time t_1, the optimum found by backtracking may not be the global one if more observations after time t_1 is added. We want the algorithm handle situations where $\tau \to \infty$.

Online Viterbi Optimization solves the situations where $\tau \to \infty$ by constructing a set of states $\mathcal{X}_t$. This set contains the global optimum q_t^* but also all other possible optimum that future observations can reveal. By letting $\mathcal{X}_{t_1}$ be the entire state space $\mathbb{S}$ and for every $\mathcal{X}_t, t < t_1$ making it smaller by taking the image under ψ_{t+1} through

$$\mathcal{X}_t = \{S_i | i = \psi_{t+1}(j) \text{ for some } S_j \in \mathcal{X}_{t+1}\}.$$

At time t_1 we have $\mathcal{X}_{t_1}$ containing all states possible. But after iterating backwards, $\mathcal{X}_t$ becomes smaller and smaller. At some time, t_2, there is only one state left in $\mathcal{X}_{t_2}$. This is the optimal state for $q_{t_2}^*$, and all other optimal states q_t^* for $t < t_2$ is given by backtracking from $q_{t_2}^*$ for their can only be exactly one state in $\mathcal{X}_t$ for $t < t_2$ because $\mathcal{X}_t \in \mathcal{X}_{t_2}$ for $t < t_2$.

5 Building the predictor

When tracking is in place and we have the path the pedestrian walked, the next step is to construct the predictor. The predictor should predict the probability that a person will enter the crosswalk for each state in $\mathbb{S}^1$.

For every position in the grid a probability is calculated for the next connected position. This probability is calculated from statistically collected tracks that update the predictor.

Only theory for constructing a predictor without history is shown here, the expansion to use history will be further explained in section 5.4.

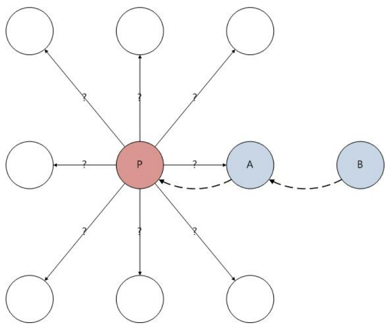


Figure 5: Showing the position (P) and possible way the person can move next. The predictor could use none, one (A) or two steps (A and B) of history.

The tracking from section 4 outputs a track

$$\mathcal{P} = (p_0, ..., p_\tau), p_i \in \mathbb{S}^1, i = 1, .., \tau,$$

a sequence of states for each person walking over the grid. Let $\mathcal{T} = (\mathcal{P}_0, .., \mathcal{P}_n)$ be the collection of all tracks from the tracker. The number of transitions from state x_i to x_j can be defined as

$$n_{i,j} = \sum_{\mathcal{T}} \sum_{\mathcal{P}} z_{i,j}(p_t, p_{t+1}),$$

where,

$$z_{i,j}(x,y) = \begin{cases} 1 & \text{if } x = x_i \text{ and } y = x_j \\ 0 & \text{otherwise} \end{cases}.$$

In the same way we define the number of visits in a state as $n_i = \sum_{\mathcal{T}} \sum_{\mathcal{P}} z_i(p_t)$, where

$$z_i(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{otherwise} \end{cases}$$

By defining the predictor function $D(x_i, x_j) = p(q_{t+1} = x_j | q_t = x_i)$, where $x_i, x_j \in \mathbb{S}^1$, the probability that person in state x_i will move to state x_j. The estimation of that probability is

$$D(x_i, x_j) = n_{i,j}/n_i.$$

5.1 Updating the predictor

It's also good to be able to update the predictor with the tracking data collected at runtime so the predictor is continuously learning. Because the predictor is keeping track of the number of samples, n_i, collected for each state, this can be used to continuously update D with new data.

The update is done by:

$$D(x_i, x_j) := \begin{cases} (D(x_i, x_j) * n_i + 1)/(n_i + 1) & \text{if } p_t = x_i \text{ and } p_{t+1} = x_j \\ D(x_i, x_j) * n_i/(n_i + 1) & \text{if } p_t = x_i \text{ and } p_{t+1} \neq x_j \end{cases}$$

$n_i := n_i + 1$
Where p_t and p_{t+1} is the current and previous position of a person given by the tracker.

5.2 Initial values

When the system is new and there is no values, predefined initial values for D is set, and set $n_i = a$ for all i. Where a is a constant depending on how fast the system should converge from the initial values. If $a = 0$ the initial values will immediately be replaced by the single data sample. If a is a large number, it will take more samples for D to converge. The predictor can for example be set to $D(\cdot) = 1/8$, meaning that all directions are equally probable.

5.3 Boundary of the grid

The event of a person leaving or entering the observed area, the data is reaching the boundary of the grid, needs special treatment in the predictor. By using the special state $\mathbb{S}^0$ and expanding the predictor with this as a position , entering and exiting of the grid can be used in the prediction. All boundary positions of the grid connect to this state.

5.4 Using the History

Using only the current position of the pedestrian will not make a good predictor, we have no way of telling from which direction he came from, and therefore

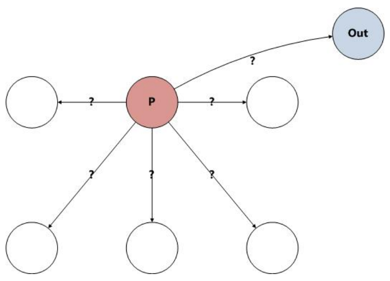


Figure 6: Showing the position (P) and possible way the person can move next. The position is on the boundary, so the person might leave the area. This is equivalent to the prediction of person going to the out position.

not be able to make a good prediction on further movement. Expanding the predictor to include more of the history is therefor required.

The predictor D is expanded using one step of history so that $D(x_h, x_i, x_j) = p(q_{t+1} = x_j | q_t = x_i, q_{t-1} = x_h)$.

The update is done in the same manner as before except that the state space has increased with one dimension. Using two steps or more is handled in more or less the same way.

The initial values can now be more refined than an equal distribution over all directions. The Values should be defined using the knowledge of in which direction the person is heading and the natural behavior of continuing in the same direction. For example as in Figure 7.

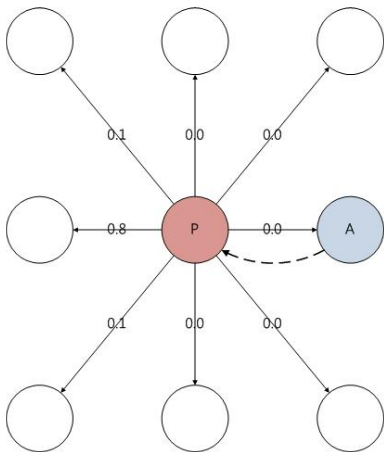


Figure 7: Showing the position (P) and possible way the person can move next.

5.5 Markov Matrix G

The predictor outputs a statistical probability where a person will move next. With this, it is also possible to calculate the probability of the person entering the crosswalk. This can be done through 3 steps.

Step 1: Constructing the Markov matrix named G is quite straight forward. A Markov matrix is the transitions of a Markov chain. A markov chain is a sequence of random variables $X_1, X_2, X_3, ...$ with the Markov property, namely that, given the present state, the future and past states are independent. Formally,

$$Pr(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = Pr(X_{n+1} = x | X_n = X_n) \text{ [4]}.$$

A Markov matrix is the representation of the transition probabilities between all states, with elements $p_{ij} = Pr(X_{n+1} = j | X_n = j)$. Every position in the grid is made a state, when the history is used, the combination of position and previous position(s) is made a state. The state that represent leaving the grid is made an absorbing state, see Figure 5.5. The Markov matrix G is then constructed from the dataset D.

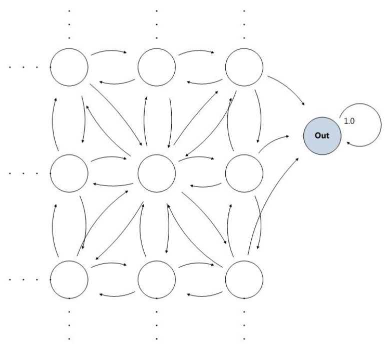


Figure 8: Here is a visualization of the markov matrix. Every node represent a state, and the arrows are the transitions available from that node. In this figure the edge of the grid is illustrated, and therefore we have the out state present.

Step 2: Now we need to decide which positions are considered located on the crosswalk. These position will be connected to a new absorbing state, which is considered to be the "Entered the crosswalk" state. The previous connections are removed. There could also be other crosswalks or pathways of interest. These could be connected to other absorbing states.

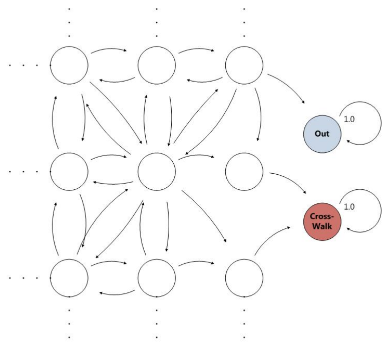


Figure 9: The states that been tagged as crosswalk will be altered so the transition to the new state.

Step 3: The altered Markov matrix G now contains the probability for an object to move from one state to the next. To get the probability of where a person will be in two steps G^2 can be used. The new absorbing state, "Entered the crosswalk", makes it possible to see what the probability is for a person in a state to get to the crosswalk in n steps, by calculating G^n. By then making a approximation of G^∞ it is possible to predict what the probability is for a person to end up on the crosswalk. To approximate G^{∞} it is enough to calculate G^{a} with a big enough. At least $a > 2 * r * c$ where r and c is the number of row and colums in the grid.

The approximated $G^{\infty}(\text{state}, \text{crosswalk})$ holds the probability that the a person in a state will and up at the crosswalk.

Let,

$$g(x) = G^{\infty}(x, \text{crosswalk}),$$

be a function that returns the probability from the matrix G^{∞} at a specific state x.

5.6 Multi person prediction

The matrix G^{∞} now gives the probability for one person ending up on the crosswalk. If there are more than one person, the interesting question is what the probability is that at least **one** of the persons enters the crosswalk. Mathematically this is described as $q = 1 - \prod(1 - p_k)$ where $p_k = G^{\infty}(\text{pos}_k, \text{crosswalk})$, is the probability that person k enters the crosswalk.

An example:
Two persons, A and B, has probabilities $p_A = 0.45$ and $p_B = 0.35$
The formula gives $1 - (1 - 0.45)(1 - 0.35) = 0.6425$, so the probability that at least one of the persons enter the crosswalk is 64.25%

6 Application

A big part of the work done on this thesis was building a working application using the theory presented in the the previous sections. The application can run with images from a live camera or loaded from a file. Images can be saved to file with the same application. The application is not tested on a real traffic light environment but is simulated on a street under a balcony.

6.1 Equipment and Software

An UEye camera with a 8mm lens was used to capture frames, this was borrowed from Smart Eye AB [11]. The native resolution is 756x480 but that is more than needed so the resolution was lowered by binning 2X both horizontally and vertically. That gave a resolution of 376x240. The camera is connected to a computer through a usb cable. A frame grabber interface was implemented to setup the camera and capture frames. The settings except binning is also auto exposure, that will give a nice white level even if there is sunlight or cloudy conditions.

As a software platform source code from Cognimatics AB [12] was used as a foundation. This was written in C code. Also the OpenCV [13] library is used, especially for camera calibration.

6.2 Camera Calibration

To get screen coordinates from world points, a calibrated camera is needed both intrinsic and extrinsic parameters. The intrinsic parameters can be found using OpenCV:s [13] camera calibration and a chessboard. The extrinsic parameters

i.e the camera position relative the world is found using relative points in the world and screen.

6.3 Collecting Data

The camera was positioned on my uncles balcony facing down toward the street. Here we could control the environment and do data collection without any major disturbances. The street did not have a crossing at that position but that is not crucial for the experiments. At first it was needed to make reference markings on the street to do the extrinsic camera calibration. Two persons where then asked to do a sequence of passes through the tracked area. The passes where made in different directions in the intersection and the video mark as well. This video was recorded to file for offline processing.

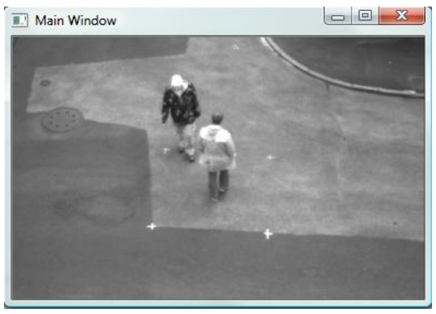
To generate more data from the collected video, it is possible to make small adjustment to the extrinsic calibration. That will make the application to see the tracked persons at another position than they originally was at. The extrinsic calibration was changed by translating the reference markings 10cm in four different directions, making it possible to run through the same recording a total of five times.

6.4 Probability Matrix G^{∞}

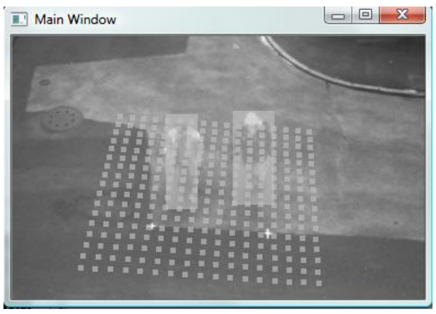
The video recorded in the previous section was then run through the tracking algorithms. The tracks updates the predictor as in section 5.1. The red points illustrated in Figure 10d are chosen to be the start of the cross walk. The probability matrix G^{∞} is created as described in section 5.5

6.5 The probability a person will enter the cross walk

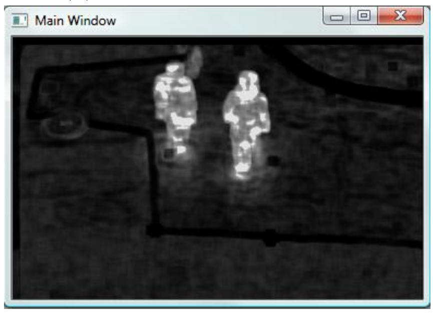
We like to calculate the probability that someone is going to the crosswalk for the current frame. By the theory in Section 5.5 this can be calculated if we have the state where the person(s) are. In the case with HMM and Online Viterbi, there is usually not a global optimum in the current frame. That is calculated later when backtracking is done. Instead by looking at the internal states $\delta_t(k)$ of the Online Viterbi algorithm a distribution can be calculated for the states. It might be possible to calculate the correct probability for each state. But in this study, only a approximation has been made. The approximation is made using a weight



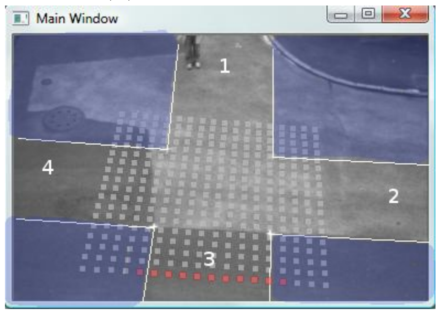
(a) Video Raw



(b) Video with grid



(c) Video Bg/Fg



(d) Video Area

Figure 10: Lägg till bättre text här

$$y_k = \frac{1}{2000}\delta_k + 1 - \frac{\delta_{\max}}{2000}$$
$$\text{if } y_k < 0 \text{ then } y_k = 0.$$

This function has been derived from empirical tests.
The approximative probability p_k for each state $\delta(k)$ is

$$p_k = \frac{y_k}{\sum y_k}.$$

When the distribution for the different states is calculated it is straightforward to use the result in Section 5.5 and get the total probability a person is going to the crosswalk. It is

$$p_{\text{crosswalk}} = \sum_k (p_k \cdot g(\text{state}_k)).$$

If there is more than one person in the state, the result from Section 5.6 is used.

6.6 Classifier

To make it easier for an application to use the probability and also to verify the result from the predictor, we classify it into three classes.

A Someone is going to the crosswalk $p_{\text{crosswalk}} > \beta$

B Uncertain $\alpha < p_{\text{crosswalk}} < \beta$

C No one is going to the crosswalk $p_{\text{crosswalk}} < \alpha$

The probability from the predictor is classified by setting thresholds α and β. Sometime the probability value is around a threshold and the classifier changes between two states rapidly. To overcome this problem hysteresis could be used. Hysteresis is when the value goes over the threshold α to enter a state, it does not go back to the other state until the value is under $\alpha - d$. In the application d is set to 0.1, see Figure 11.

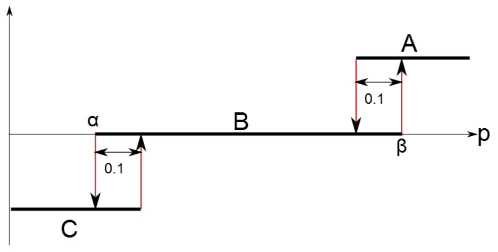


Figure 11: By using hysteresis, quick changes of from states A,B and C can be avoided when probability p changes. α and β is the thresholds for the classifications.

7 Results

The results and figures shown in this section is from my application based on the data I collected.

7.1 Background/Foreground Segmentation

A sample from the background foreground segmentation is shown in Figure 12. Here two persons are walking and the resulting image is clearly brighter around them, indicating they belong to the foreground.

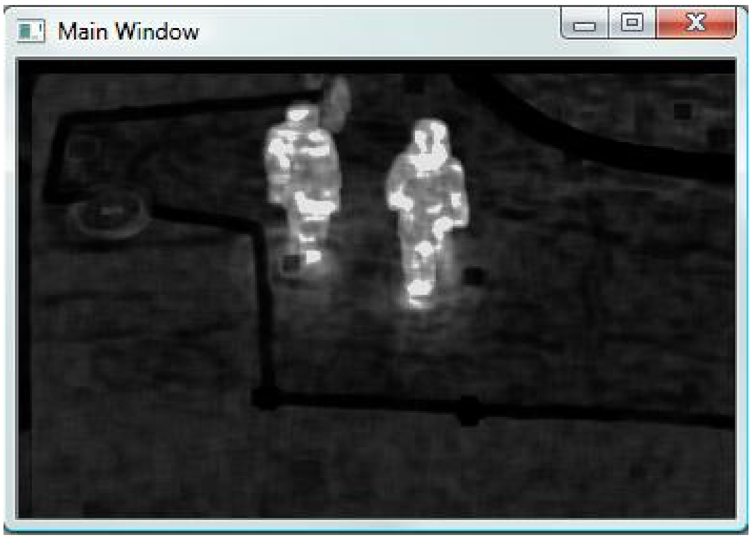


Figure 12: Background foreground segmentation.

7.2 Tracking

Figure 13 shows a sample from tracking. The illustrated walk trails are not the optimum path calculated by the tracker, instead it is the path with the highest likelihood at that frame.

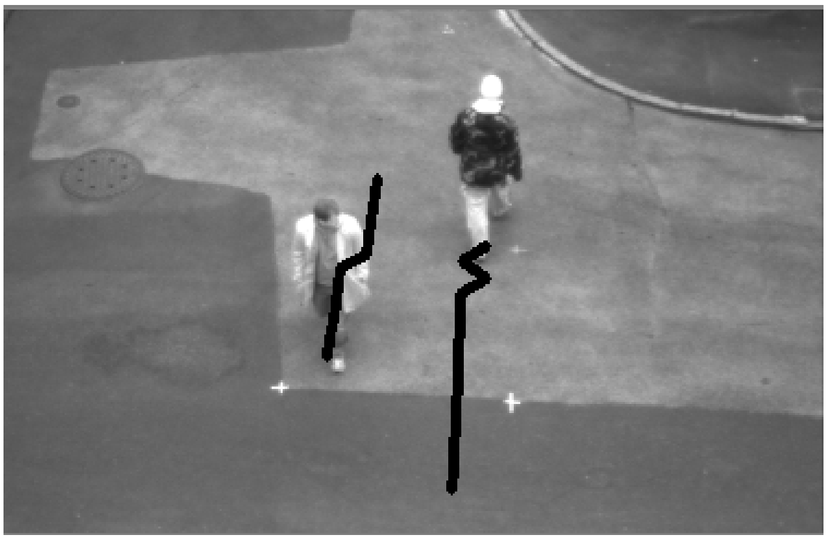


Figure 13: Walk trails.

7.3 One direction Setup

This setup is meant the simulate a straight pathway to the crosswalk as illustrated in Figure 14. The red squares are marked as crosswalk.

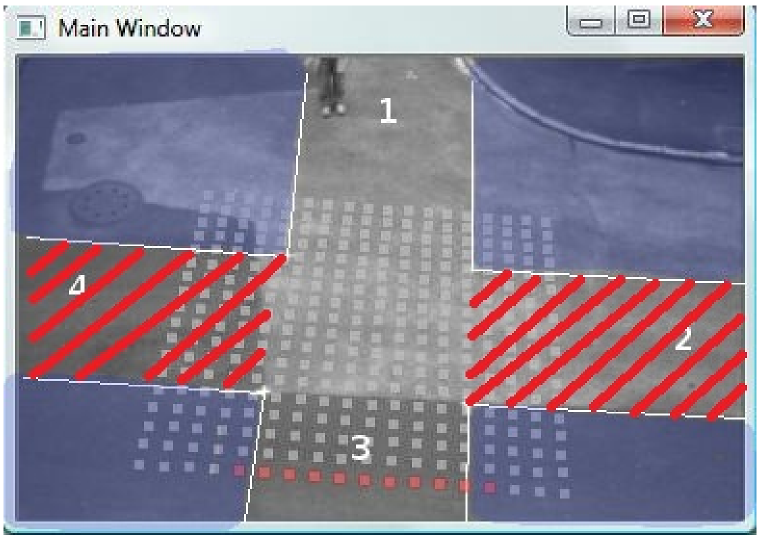


Figure 14: Included areas in the data set.

7.3.1 Data collection

Data was collected for a predictor not using history and one that uses history of one step. Using further history was unfortunately not possible due to lack of the large amount of training data required.

In Figure 15 and 16 the number of collected data samples per state is illustrated.

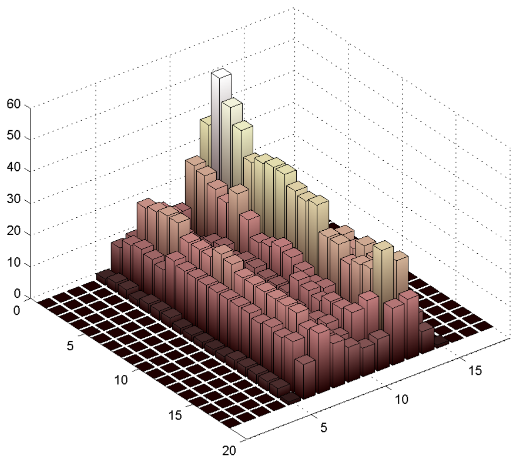


Figure 15: Collected data samples per state when not using history. The crosswalk is located in the top left.

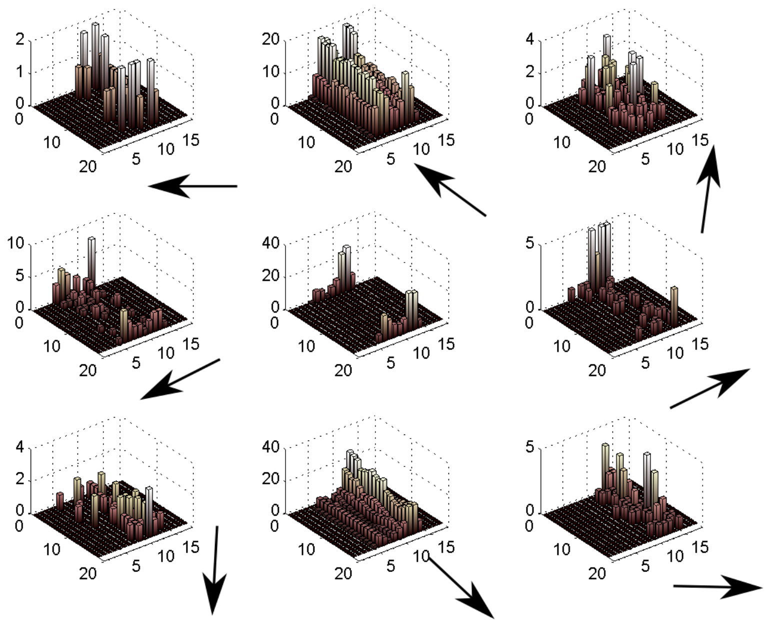


Figure 16: Collected data samples per state using history of one position. The arrows indicate the direction of movement. The center chart is representing entry on the boundary. The crosswalk is located in the top left.

7.3.2 Prediction

The resulting predictor that doesn't use history, Figure 17, does not give any promising results as expected. Due to the lack of history no good prediction can be done. The only interesting to see is that the majority of trails to the crosswalk has been made on one side and on the opposite side, away from the crosswalk.

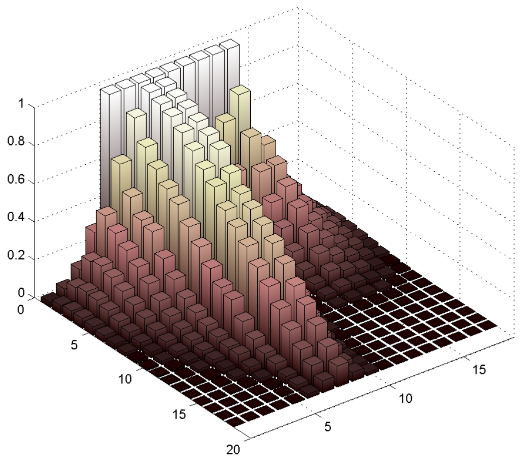


Figure 17: Probability to enter crosswalk when not using history. The crosswalk is located in the top left.

The predictor that uses history of one, shows much more promising results. In Figure 18 we can se that a person moving in the direction of the crosswalk is predicted with a high probability to end up on the crosswalk. In the same way a person moving away from the crosswalk is predicted with a low probability to end up on the crosswalk.

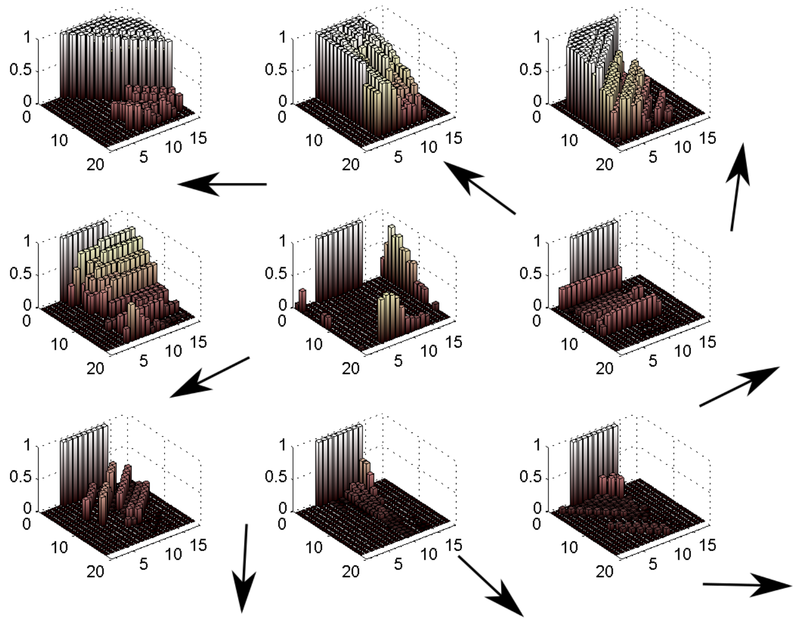


Figure 18: Probability to enter crosswalk when using history of one position. The arrows indicate the direction of movement. Center plot is representing entry on the boundary. The crosswalk is located in the top left.

7.3.3 Validation

I will only do validation on the predictor that uses history.

To validate the predictor five recordings where made on normal and tricky situations. These where hand annotated with different states, like, "Person in area that will end up on crosswalk", "Person in area that will not end up on crosswalk" and the combined state. In Figure 19 is the result from one of these tests. In the illustration we can see that a person is approaching the crosswalk and the predictor immediately gives a high probability that the person is entering the crosswalk. The person then stops to wait for green light. As he stops he leans back witch fools the tracker to believe he is moving in the opposite direction which can be seen in the figure that the probability decreases.

When the first person has left the tracked area, a new person enters from the crosswalk. In the figure the probability is very high at first. This is because the person is actually standing on the crosswalk states in the grid. When the person is moving further the probability decreases to a very low level.

If we use the classifier from section 6.6 and compare the result with the hand annotated test recordings. We can calculate the true positive and false positive classifications from the classifier. The results of classifying "Someone is going to the crosswalk" is shown in Figure 20. We can see that there is a gap between the true positive and the false positive. If 0.5 is chosen as the threshold the classifier was able to correctly classify 70% of the time a person walk to the crosswalk in the tracked area, and incorrectly classified 5% of the time when a

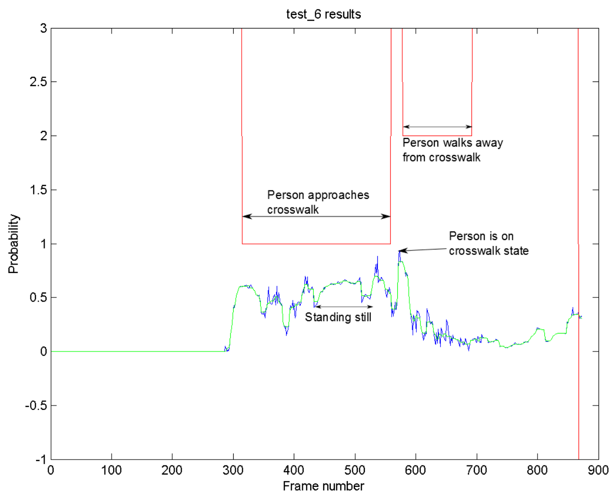


Figure 19: Test sequence where a person approaches the crosswalk than another person comes from the crosswalk and walks away from it.

person was not walking to the crosswalk as such.

Figure 21 shows the result of the classification "No one approaches crosswalk". Here we can see that if we set the threshold to 0.3 we would have correctly classified more than 90% correctly when there is a person in the area not walking to the crosswalk. At the same threshold unfortunately it classifies 15% wrong i.e. that there is no one approaching the crosswalk in the area when it actually is.

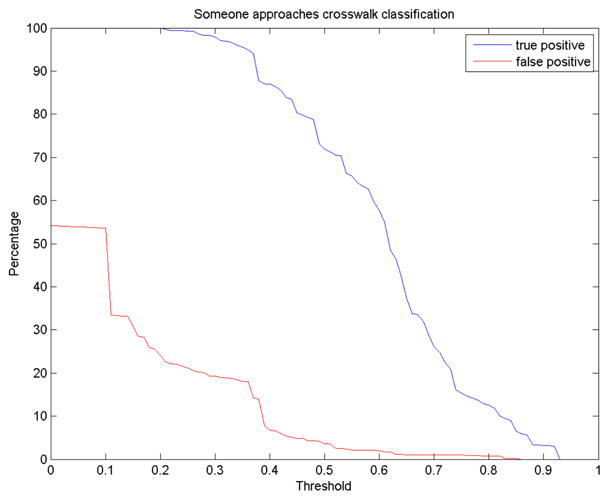


Figure 20: "Someone approaches" classification results relative the threshold.

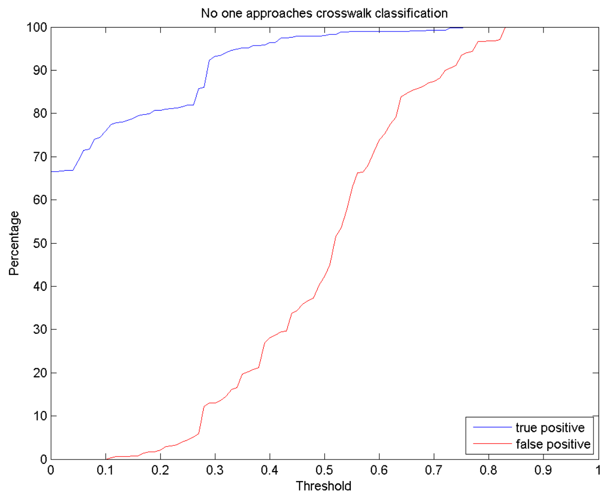


Figure 21: "No one approaches" classification results relative the threshold.

7.4 Multiple Direction Setup

In this setup we simulate a four-way crossing with a crosswalk in one direction as illustrated in Figure 22. The red squares are marked as crosswalk.

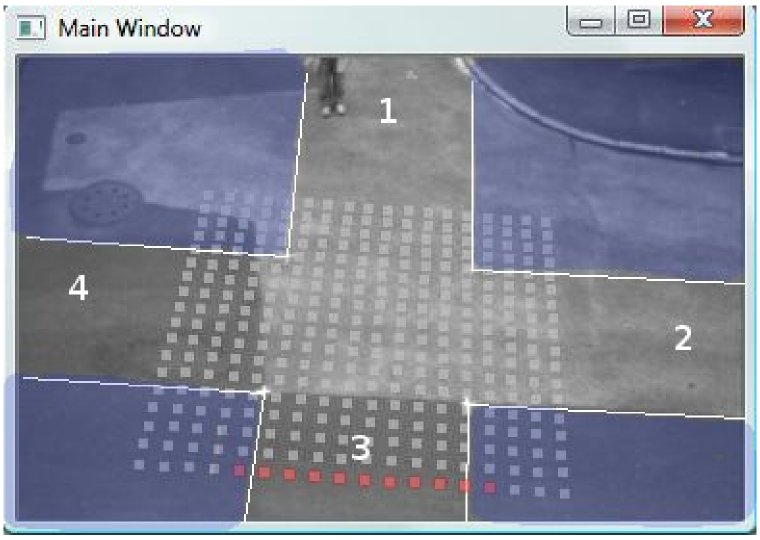


Figure 22: Included areas in the data set.

7.4.1 Data collection

Data was collected for a predictor not using history and one that uses history of one step. Using further history was unfortunately not possible due to lack of the large amount of training data requried.

In Figure 23 and 24 the number of collected data samples per state is illustrated.

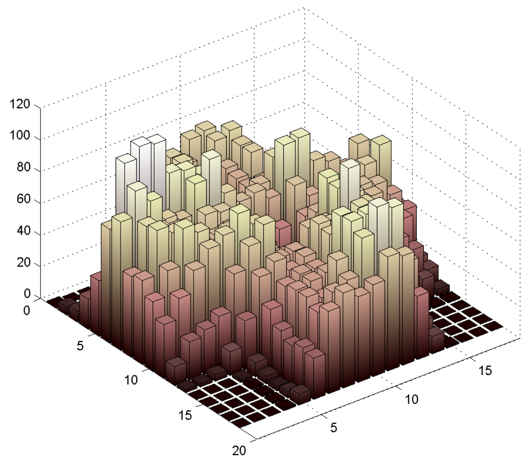


Figure 23: Collected data samples per state when not using history. The crosswalk is located in the top left.

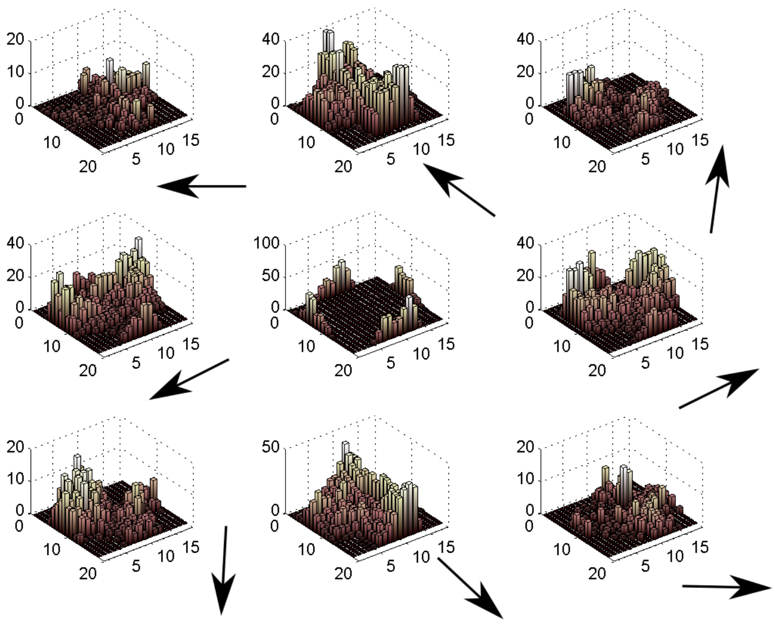


Figure 24: Collected data samples per state when using history of one position. The arrows indicate the direction of movement. Center plot is representing entry on the boundary. The crosswalk is located in the top left.

7.4.2 Prediction

As earlier in the case with the straight direction,section 7.3, using prediction without history is pointless as illustrated in Figure 25. In the case with history of length one as shown in Figure 26, prediction is possible. The probability of the predictor will not get high until the person has reach a point that turning in another direction is no longer a feasible option. This is expected because the data collected had an even spread in the directions the persons walked.

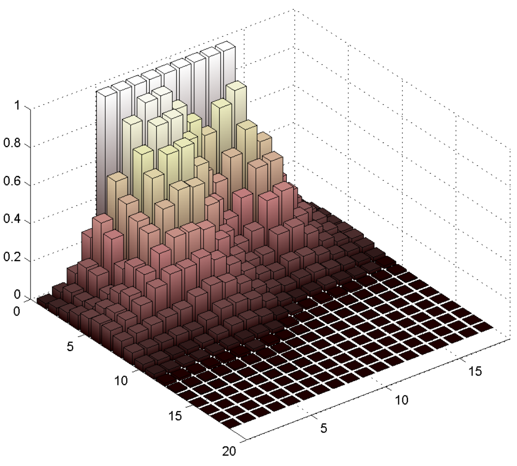


Figure 25: Probability to enter crosswalk when not using history. The crosswalk is located in the top left.

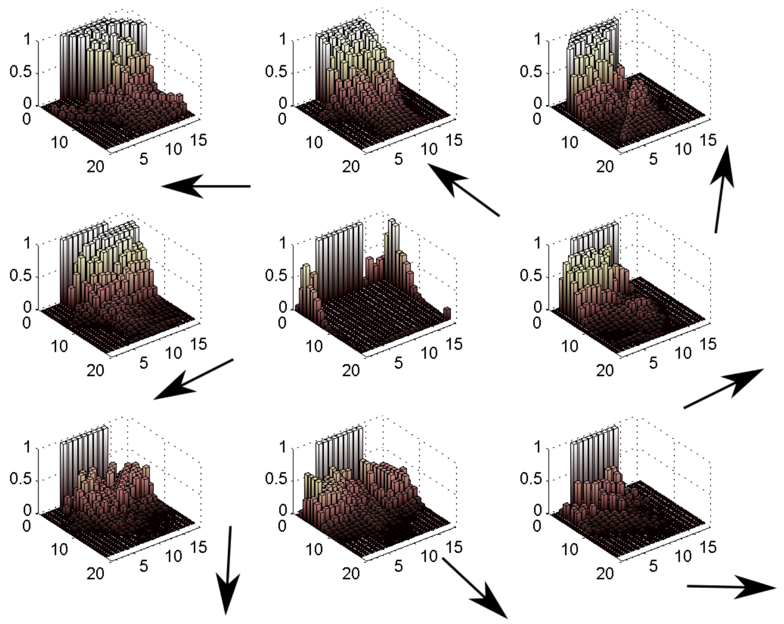


Figure 26: Probability to enter crosswalk when using history of one position. The arrows indicate the direction of movement. Center chart is representing entry on the boundary. The crosswalk is located in the top left.

7.4.3 Validation

Because the predictor not using history will not perform well, only vaidation on the predictor that uses history is done.

To validate the predictor 12 recordings where made on normal and tricky situations. These where hand annotated with different states, like, "Person in area that will end up on crosswalk", "Person in area that will not end up on crosswalk" and the combined state. In Figure 27 the result from the same test as in the straight case shown in Figure 19. The probability the predictor produces is very similar to the one from the straight case except from the first part where the probability is lower. This is because the predictor can not decide which direction the person is going until he passes the intersection.

Figure 28 shows a sequence where first a person walks from the top way, path 1 in Figure 22, and walks to the crosswalk. After a person walks from the side, path 2, and walk straight to the other side and exits at path 4. The probability result from the predictor is similar to the previous one except the probability is low for the second person, which is correct as the person never near the crosswalk.

If we use the classifier from section 6.6 and compare the result with the hand annotated test recordings. We can calculate the true positive and false positive classifications from the classifier. The results of classifying "Someone is going to the crosswalk" is shown in Figure 29. We can see that there is a gap between the true positive and the false positive. If 0.5 is chosen as the threshold the

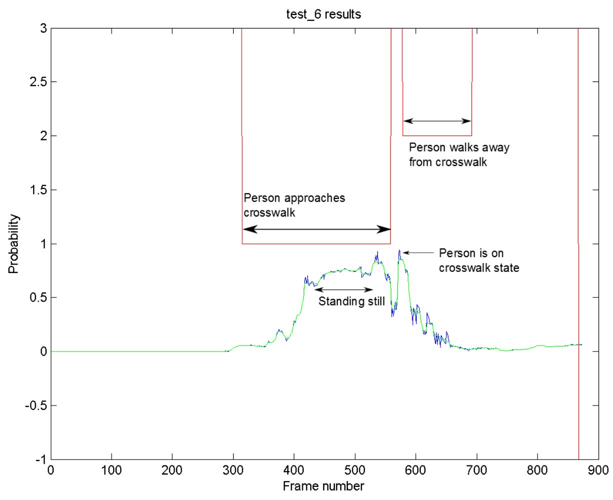


Figure 27: Test sequence where a person approaches the crosswalk than another person comes from the crosswalk and walks away from it.

classifier was able to correctly classify 55% of the time a person walk to the crosswalk in the tracked area, and incorrectly classified 2% of the time when a person was not walking to the crosswalk as such. Comparing to the straight case this result is much lower but can be explained by the insecurity the predictor has before the person change direction.

Figure 30 shows the result of the classification "No one approaches crosswalk". Here we can see that if we choose the threshold 0.1 we would have correctly classified more than 85% correctly when there is a person in the area not walking to the crosswalk. At the same threshold unfortunately it classifies 25% wrong i.e. that there is no one approaching the crosswalk in the area when it actually is. The false positive is larger than in straight case. That is because it is hard for the predictor to predict if the person is going to the crosswalk or not until the person has passed the intersection.

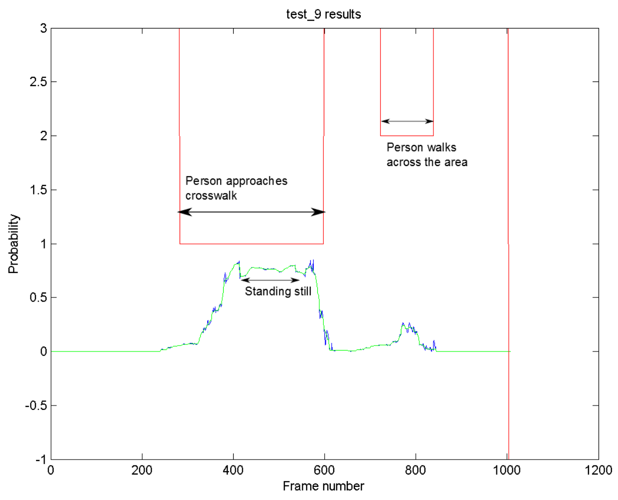


Figure 28: "Someone approaches" classification results relative the threshold.

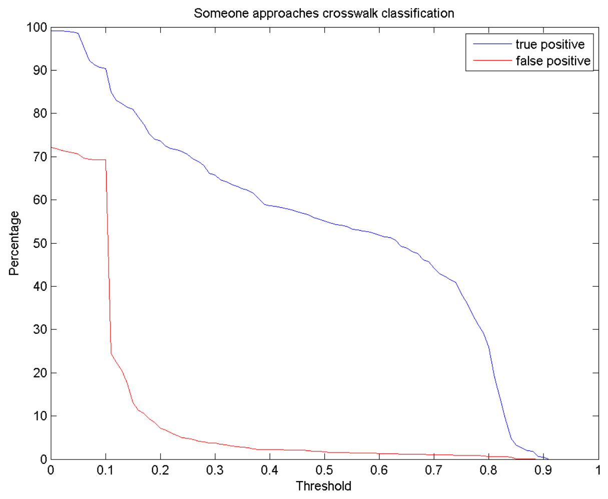


Figure 29: "Someone approaches" classification results relative the threshold.

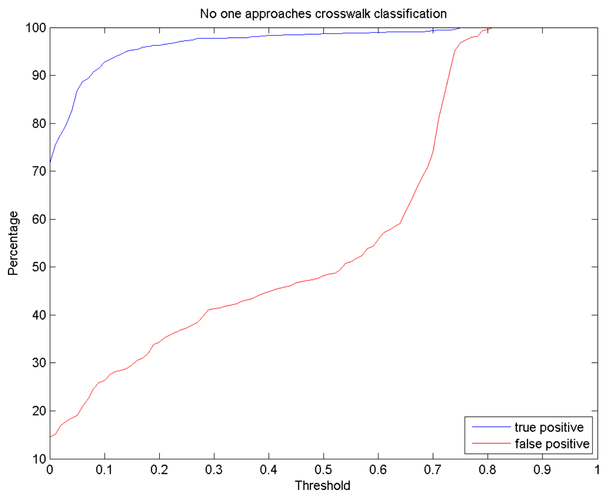


Figure 30: "No one approaches" classification results relative the threshold.

8 Discussion/Improvements

The development of this thesis started out with trying to implement the algorithms on a embedded platform. This was very time consuming and slowed down development time. The work was not really related to the thesis and a decision was made to skip the embedded platform and make the development on a PC instead. A lesson learned is to start of with a PC based solution and then port to embedded.

8.1 Background/Foreground Segmentation

The chosen bg/fg algorithm uses cross correlation features. It is independent on global intensity changes and robust for small objects which is good for outdoor scenes. Sometimes the middle of a persons jacket correlate well with the background resulting in a low probability for foreground. This did not cause any problem because the edges of the jacket did not correlate with the background and resulted in a high probability and the total was large enough to be trackable.

8.2 Tracking

To track the pedestrians, Hidden Markov Model was used. It worked well and was really robust. In this thesis a rather sparse grid was used. That made the tracked trails jagged, see Figure 13. Increasing the resolution of the grid makes the tracking more precise and not as jagged as in this thesis but it also increases the requirements of processing power and memory. The predictor also needs more data before converging.

The optimal camera placement is from a birds perspective, but if this is not possible and a more horizontal view is required, problems with occlusion can occur. However the HMM tracking worked very well in this regard, except when the person was occluded the whole way through the tracked area.

8.3 Predictor

The predictor is a novel algorithm based on a Markov theory. The advantage is that the lookup in the predictor is fast but a predictor that looks at a longer history walk trails requires a lot of memory. The memory grows with a factor eight for every history point.

The update algorithm of the predictor requires a lot of computational power, but it can be done during night time or in a separate thread in the software.

By increasing the length of the history the predictor should be able to perform better. But in this thesis it was not possible to test due to the large amount of data needed for training and validation. For each step of history added, the amount of data needed for the predictor to converge grows as the memory, by a factor 8. If a very large history is used, the predictor will be using the complete track from the moment the person enters.

Another possible improvement to the predictor is to increase the resolution of the grid. This has not been investigated in this thesis, but a higher resolution should increase precision in the predictor. If the distance between grid points is halved, the necessary data collection grows with a factor of four.

The initial values are not that important thus they are phased out with time. If it is possible to make a good assumption of the initial values the predictor would work directly, without collecting a lot of data.

Initially when a person enters the area,the probability the predictor gives is based on a approximation of the statistical distribution of persons going to the crosswalk or not from that position.

8.4 Results

The validation is made for different thresholds on the classifier. Depending on the application different behavior on the classifier is wanted. If a lower threshold is chosen for the classification "No one is going to the crosswalk" we get less false positives but that also leads to less true positives.

The validation for "No one is going to the crosswalk" only included frames when there was a person in the tracked area. If frames would be included that did not have any persons visible, the rate for true positives would increase a lot because the predictor outputs near to zero probability when there is no person in the area.

The results shows that the algorithm works, but is not as robust as wanted. When a persons stops and waits for green light he tends to have some backwards movement. Given that we only used a history of one it fools the predictor that he started to move away from the crosswalk. A solution to this is described in section 8.5.

8.5 Improvements

Instead of just using some grid points on the edge as the crosswalk, using a area around the crosswalk would give more robust result. This would increase the true positive when standing and waiting for green light.

Some simple logics can be added to the classification, for example, persons that enters from the crosswalk are probably not going to it and should not be predicted as such.

By looking at velocity the predictor could estimate when the person will reach the crosswalk.

9 Conclusions

The purpuse of this thesis was to create a detector to decide if a person is entering the crosswalk or not. With the application developed it has been showed as a "proof of concept". Using the novel predictor to detect persons "entering the crosswalk" we get

Setup	@ p Threshold	% true positive	% false positive
Straight	0.5	70%	5%
Four way	0.5	55%	2%

and for detection of someone is "not entering the crosswalk" when there is a person in the area

Setup	@ p Threshold	% true positive	% false positive
Straight	0.3	90%	15%
Four way	0.1	85%	25%

There is a large room for further improvements and given that the predictor has so large percentage of false positives, improvements needs to be made.

Acknowledgments

References

[1] Håkan Ardö *Multi-Target Tracking Using On-line Viterbi Optimisation and Stochastic Modelling* 2009.

[2] P. Viola and M. Jones *Rapid object detection using a boosted cascado of simple feauters.* 2001: Random House, N.Y.

[3] L.R. Rabiner *A tutorial on hidden markov models and selectes applications in speech recognition.* Proc. IEEE, 77(2):257-286, 1989.

[4] Markov chain, Wolfram MathWorld `http://mathworld.wolfram.com/MarkovChain.html`

[5] Gabriel J.Borstow and Roberto Cipolla *Unsupervised Bayesian Detection of Independent Motion in Crowds.* IEEE CVPR 2006: New York

[6] Dima Damen, David Hogg *Recognizing Linked Events: Searching the Space of Feasible Explanations.* School of Computing, University of Leeds

[7] Christopher James Needham *Tracking and Modelling of Team Game Interactions.* 2003

[8] B Benfold and I D Reid *Stable Multi-Target Tracking in Real-Time Surveillance Video.* Colorado Springs, June 2011

[9] Neil Johnson and David Hogg *Learning the Distribution of Object Trajectories for Event Recognition.* School of Computer Studies, The University of Leeds

[10] Baumberg A. and Hogg D *An efficient method for contour tracking using active shape models.* In IEEE Workshop on Motion of Non-rigid and Articulated Objects, pages 194–199. IEEE Computer Society Press, November 1994. IEEE Catalog No. 94TH0671-8.

[11] Smart Eye AB `http://www.smarteye.se`

[12] Cognimatics `http://www.cognimatics.com`

[13] OpenCV `http://opencv.org/`

A Infinite Viterbi algorithm

Infinite Viterbi algorithm is listed in Algorithm 1. The diffrence from classic viterbi is that it only stores the m largest $\delta(i)$. The rest of the states will have a upper bound $\delta_{\max}(i)$ for the rest. If m is large enough the global optimum will be found by backtracking. This algorithm will decide if it was the case or not. If it was not it can be re-run with a larger m.

The algorithm is divided in three parts. The first part (lines 1-4) initiates, second part (lines 5-24) is forward propagation and the final part (lines 25-35) do the backtracking.

Initialization (1-4):
Line 1 sets $\tilde{\delta_0}(i) = \pi_i b_{i,0}$ which is the starting sequence probability's. Line 2-4 creates $\hat{\delta_0}$ from the m largest $\tilde{\delta_0}$ sorted as a decreasing function. It also stores the permutation h_0 that makes it possible to backtrack. The upperbound for all discarded states is saved as $\delta_{\max}(0)$.

Propagation (5-24):
This will be executed for all $0 < t < \tau$. The first thing to do (line 6) is to use the reachable function that is defined as $R(\hat{\mathcal{S}}) = \{i | a_{i,j} > 0 \text{ for some } j \in \hat{\mathcal{S}}\}$. This function will give all the states that are reachable from the set $\hat{\mathcal{S}}$. For each of the states that are reachable (line 7) calculate which previous state it is most likely to come from and create temporary δ_{stored}. Also store which state it came from in $j_{\max}$ (line 8). To make sure that there is no previusly discarded state that might give a higher likelihood calculate $\delta_{\text{discarded}}$ (line 9) which is the largest probability from a discarded state. Here is the constant $a_{\max}$ used. It is defined as $a_{\max} = \max_{i,j} a_{i,j}$, which is the largest transitional probability possible. If δ_{stored} is bigger than $\delta_{\text{discarded}}$ (line 10) we store $\tilde{\delta_t}(i)$ and $\hat{\psi_t}(i)$ (line 11-12). If it's not (line 13) we set $\hat{\psi_t}(i) = -1$ and $\tilde{\delta_t}(i) = \delta_{\text{discarded}}$(line 14-15) this will indicate in backtracking that we don't know the optimum path. When all reaching states are calculated, the next step is to sort $\tilde{\delta_t}$ and store the m largest in $\hat{\delta_t}$ and the permutations in h_t (line 18 and 23). Then to decide δ_{maxcalc}, the biggest discarded $\hat{\delta_t}$, and δ_{maxdisc} which comes from previous discarded states $\delta_{\max}(t-1)$. Save the maximum of δ_{maxdisc} and δ_{maxcalc} to $\delta_{\max}(t)$ and do the next iteration.

BackTracking (25-35):
Start the backtracking by extracting the best state from $\hat{\delta_\tau}$ (line 25). If the value is lower then the upper bound of the discarded states (line 26) the optimum can not be found. Otherwise continue with the backtracking (line 29-35) and look up which sequence of states gives the highest probability through going backwards in ψ. If ψ gives -1 anytime during backtrack the optimum could not be found. Using the permutation h_t the optimal sequence $\tilde{q}_t, 0 < t < \tau$ is created (line 34) and returned.

Algorithm 1 Infinite Viterbi

```
1: $\tilde{\delta}_0(i) = \pi_i b_{i,0}$
2: $(\hat{\delta}_0, h_0) = \text{sort}(\tilde{\delta}_0)$
3: $\delta_{\max}(0) = \max_{i>m}(\hat{\delta}(i))$
4: Discard $\hat{\delta}_0(i)$ for $i > m$
5: for all $t = 1$ to $\tau$ do
6:     $\mathcal{S} = \mathcal{R}(\{h_{t-1}(i) | i = 1, ..., m\})$
7:     for all $i \in \mathcal{S}$ do
8:         $(\delta_{\text{stored}}, j_{\max}) = \max_{1 \le j \le m}(\hat{\delta}_{t-1}(j) a_{h_{t-1}(j),i})$
9:         $\delta_{\text{discarded}} = \delta_{\max}(t-1) a_{\max}$
10:        if $\delta_{\text{stored}} > \delta_{\text{discarded}}$ then
11:            $\tilde{\delta}_t(i) = \delta_{\text{stored}} b_{i,t}$
12:            $\hat{\psi}_t(i) = j_{\max}$
13:        else
14:            $\tilde{\delta}_t(i) = \delta_{\text{discarded}} b_{i,t}$
15:            $\hat{\psi}_t(i) = -1$
16:        end if
17:    end for
18:    $(\hat{\delta}_t, h_t) = \text{sort}(\tilde{\delta}_t)$
19:    Find some $b_{\max} \ge b_{i,t}$ for all $i \notin \mathcal{S}$
20:    $\delta_{\text{maxcalc}} = \max_{i>m | h_t(i) \in \mathcal{S}} \hat{\delta}_t(i)$
21:    $\delta_{\text{maxdisc}} = \delta_{\max}(t-1) a_{\max} b_{\max}$
22:    $\delta_{\max}(t) = \max(\delta_{\text{maxcalc}}, \delta_{\text{maxdisc}})$
23:    Discard $\hat{\delta}_t(i)$ for $i > m$
24: end for
25: $\hat{q}_\tau = \text{argmax}_{i \le m}(\hat{\delta}_\tau(i))$
26: if $\hat{\delta}_\tau(\hat{q}_\tau) \le \delta_{\max}(\tau)$ then
27:    return Optimum not found, retry with larger $m$
28: end if
29: for $t = \tau - 1$ to 0 do
30:    $\hat{q}_t = \hat{\psi}_{t+1}(\hat{q}_{t+1})$
31:    if $\hat{q}_t == -1$ then
32:        return Optimum not found, retry with larger $m$
33:    end if
34:    $\tilde{q}_t = h_t(\hat{q}_t)$
35: end for
36: return Global optimum found!
```

Master’s Theses in Mathematical Sciences 2013:E22
ISSN 1404-6342

LUTFMA-3245-2013

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/