A Viterbi-like Procedure for 2D-Data Modelling

Stéphane Marchand-Maillet - Multimedia Communications
Email: Stephane.Marchand@Eurecom.fr
Phone: +33 (0)4.93.00.26.79 - Fax: +33 (0)4.93.00.26.27

Date: February 8, 2000
Technical Report RR-99-052
Confidential: No

Eurécom's research is partially supported by its industrial members: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

EURECOM
Multimedia Communications

Institut EURECOM – BP 193.
06904 Sophia Antipolis – France
T.R. RR-99-052 – February 8, 2000

A Viterbi-like Procedure for 2D-Data Modelling

Stéphane Marchand-Maillet

Stephane.Marchand@Eurecom.fr
http://www.eurecom.fr/~marchand/

Institut EURECOM
Department of Multimedia Communications
Technical Report RR-99-052
February 8, 2000

Abstract

This report details the construction of a Viterbi-like procedure that is able to handle two-dimensional data such as images. The aim is to associate the most likely structure (2D-state sequence) to an image, given specific model parameters.

Based on the knowledge of existing 1D- and pseudo-2D (P2D) Viterbi procedures, we detail the theoretical foundations of our new procedure. The algorithm is then given and approximations made during the developments analysed. This algorithm is evaluated for efficiency (estimation of computational and memory loads) and applied to real case studies. The problem of training is addressed in this report and results are presented.

This report further presents extensions that can be made in a way to chain the models together in order to obtain a meta-segmentation which would remove the need for a pre-segmentation step and, as in speech processing, embed this step into a consistent procedure.

This report should give a good overview of the capabilities of 1D-, P2D- and 2D-Viterbi procedures. The literature cited in reference should also constitute a good starting point for the reader interested in detailing one aspect or the other.

Résumé

Ce rapport détaille la construction d'une procédure de type Viterbi capable de s'appliquer à des données bi-dimensionnelles telles que les images. Le but est d'associer la structure (séquence d'états bi-dimensionnelle) la plus probable à une image à partir des paramètres d'un modèle donné.

Sur la base de l'expérience donnée par les procédures de Viterbi mono-dimensionnelles et pseudo-bidimensionnelles, une première partie de ce rapport présente les fondements théoriques de notre méthode. L'algorithme associé est alors détaillé et analysé en termes des approximations faites et de la complexité (calcul et mémoire). Une deuxième partie présente les applications de cette technique à des données réelles et détaille le problème de l'entraînement des modèles.

Ce rapport détaille aussi les extensions qui peuvent être faites de ces modèles à des "meta"-modèles qui permettraient d'inclure la procédure de pré-segmentation des données dans une procédure unique opérant la segmentation à tous les niveaux d'une façon cohérente.

Ce rapport devrait fournir un survol complet des procédures de Viterbi et la littérature citée en références devrait constituer un point de départ utile à celui qui désire développer un aspect précis de ce type de procédures.

Keywords: Image analysis, Video analysis, Stochastic Modelling, Viterbi algorithm

Stéphane Marchand-Maillet. *A Viterbi-like Procedure for 2D-Data Modelling.* Tech. Rep. RR-99-052, Institut EURECOM, Multimedia Communications, February 8, 2000.

Contents

Introduction 1

1 Theoretical details 2
- 1.1 1D Viterbi Procedure . . . 2
  - 1.1.1 Notation . . . 2
  - 1.1.2 Development . . . 3
- 1.2 Pseudo-2D Viterbi Procedure . . . 3
  - 1.2.1 Notation . . . 3
  - 1.2.2 Developments . . . 4
- 1.3 A novel 2D Viterbi procedure . . . 5
  - 1.3.1 Principle . . . 5
  - 1.3.2 Developments . . . 6
  - 1.3.3 2D Viterbi procedure . . . 8
  - 1.3.4 Analysis . . . 11
- 1.4 Chaining the 2D models . . . 13
  - 1.4.1 Notation . . . 14
  - 1.4.2 Inner-model non-null state transition . . . 14
  - 1.4.3 Transition from a null state . . . 15
  - 1.4.4 Transition to a null state . . . 15
  - 1.4.5 Using one unique global model . . . 16
  - 1.4.6 Null state sequences . . . 17

2 Applications 18
- 2.1 Practical implementation . . . 18
- 2.2 Single model structure: Binary image restoration . . . 21
  - 2.2.1 Salt and pepper noise . . . 21
  - 2.2.2 Gaussian blur . . . 21
  - 2.2.3 Model deformation . . . 21
- 2.3 Single model structure: Recognition of shapes . . . 22

A Configuration files 26

B The 2D data modelling procedure 31
- B.1 Description . . . 31
- B.2 Command summary . . . 33
  - B.2.1 General . . . 33
  - B.2.2 Parameters of the execution . . . 33
  - B.2.3 Management of the model files . . . 33
  - B.2.4 Calculations . . . 34
- B.3 Notes . . . 35

Bibliography 36

Introduction

In this report, we detail the construction of a stochastic modelling technique for 2D colour image analysis. In [6–8], the use of stochastic modelling was introduced when detailing adapted Hidden Markov Models for image analysis. It was shown that these tools allowed for flexibility in designing image models. However, these models also exhibited a lack of geometrical consistency which generated problems for creating 'semi-rigid' models. This was due to the fact that the two-dimensional structure of the image was taken at two successive levels (*i.e.*, pixel and line levels).

The aim of the present study is to extend the Viterbi procedure, known as efficient to find a best path within a trellis, for handling two-dimensional data while keeping a reasonable computational load. Chapter 1 introduces the concept and theory of a two-dimensional Viterbi procedure and Chapter 2 proposes to illustrate the use of this procedure in the context of face detection and localisation in colour images extracted from generic video sequences.

Chapter 1

Theoretical details

The Viterbi procedure is a dynamic programming procedure where the aim is to associate a structure with a given set of observations. This section gradually introduces this type of technique and explains major issues about its extension to two-dimensional data. It concludes with the detailed presentation of the novel 2D procedure that we aim at developing and testing in this report.

1.1 1D Viterbi Procedure

1.1.1 Notation

Without any loss of generality, we consider a finite set of observations $O = \{o_1, \ldots, o_T\}$ (from time $t = 1$ to $t = T$). Each observation is supposed to be taken from a (finite or infinite) vocabulary $\mathcal{V}$ composed of "letters" v_l (*i.e.*, $\forall\, t\ \exists l$ such that $o_t = v_l$). Given a finite state machine (FSM) λ with N states s_i, $i = 1, \ldots, N$, at each time step t, the FSM is in a given state $q_t = s_i$[1]. At $t = 1$, the system is in a given state s_i with the initial state probability $\pi_i = P[q_1 = s_i]$ and state changes between states s_i and s_j occur with the transition probability $a_{ij} = P[q_t = s_j | q_{t-1} = s_i]$ (considered as time-independent). Any letter v_l can be an output of state s_i with the probability $b_i(v_l) = P[o_t = v_l | q_t = s_i]$ (also considered as time-independent). Graphically, the FSM can be represented as in Figure 1.1. The link to Hidden Markov Modelling is trivial here (see *e.g.*, [1, 6, 11, 12]).

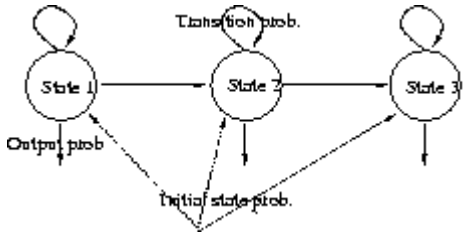

Figure 1.1: A one-dimensional FSM.

The aim is then to find the most likely state sequence $Q^* = \{q_1^*, \ldots, q_T^*\}$ corresponding to O. There may be a number of different state sequences $Q = \{q_1, \ldots, q_T\}$ which allow for outputting O. Q^* is said to be the most likely since it maximises the value

$$P[O|Q, \lambda]P[Q|\lambda] = \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \ldots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

.

[1] Note that we will sometimes use the notation "$q_t = i$" for "$q_t = s_i$".

1.1.2 Development

At each time step t we calculate the value,

$$\delta_t(i) = \max_{\{q_1,\ldots,q_{t-1}\}} P[q_1,\ldots,q_{t-1},q_t=s_i,o_1,\ldots,o_t|\lambda]$$

for every state s_i in the model. By definition,

$$\delta_1(i) = \pi_i b_i(o_1) \quad 1 \le i \le N$$

And, by induction,

$$\delta_{t+1}(i) = \max_{j=1\ldots N}[\delta_t(j)a_{ji}]b_i(o_{t+1}) \tag{1.1}$$

The maximal probability will therefore be given by,

$$P[O|\lambda] = \max_{i=1\ldots N} \delta_T(i)$$

In order to be able to retrieve the sequence of states Q^* leading to this value, the best path generated by this maximisation is stored in the array $\phi_t(i)$.

The complete 1D Viterbi (forward) procedure is therefore given as follows.

1. $\delta_1(i) = \pi_i b_i(o_1) \;\; 1 \le i \le N$
2. $\delta_t(i) = b_i(o_t)\max\{\delta_{t-1}(j)a_{ji} \;;\; j=1\ldots N\} \;\; 1 \le i \le N \;\; 2 \le t \le T$
3. $\phi_t(i) = \arg\max\{\delta_{t-1}(j)a_{ji} \;;\; j=1\ldots N\} \;\; 1 \le i \le N \;\; 2 \le t \le T$
4. $P[O|\lambda] = \max\{\delta_T(i) \;;\; i=1\ldots N\}$
5. $q_T^* = \arg\max\{\delta_T(i) \;;\; i=1\ldots N\}$

The best state sequence Q^* is then retrieved by (backward procedure),

$$q_t^* = \phi(q_{t+1}^*) \qquad t = T-1,\cdots,1$$

1.2 Pseudo-2D Viterbi Procedure

1.2.1 Notation

The aim of this section is to generalise the previous one-dimensional models in order to handle two-dimensional data. Typically, causality is introduced in two-dimensional data by defining a dependency between neighbouring items. For instance, consider a set of data O organised in a matrix-like manner

$$O = \begin{pmatrix} o_{11} & \cdots & o_{x1} & \cdots & o_{X1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{1y} & \cdots & o_{xy} & \cdots & o_{Xy} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{1Y} & \cdots & o_{xY} & \cdots & o_{XY} \end{pmatrix}.$$

Causality can be expressed as the dependency between an item o_{xy} and its neighbours in a line-scanning order. For example,

$$\begin{array}{ccccc} o_{x-1y-1} & & o_{xy-1} & & o_{x+1y-1} \\ & \searrow & \downarrow & \swarrow & \\ o_{x-1y} & \rightarrow & o_{xy} & & \end{array}$$

However, a fully connected two-dimensional structure for a model leads to exponential complexity problems for recovering the structure of a 2D observation sequence [4]. The approach taken here is therefore to introduce two-dimensional causality at the line level. In other words, a line $O_y = \{o_{1y},\ldots,o_{Xy}\}$ is now considered as an observation and the sequence $O = \{O_1,\ldots,O_Y\}$

is also modelled by an upper-level model, just as every line has been modelled by a 1D-model in Section 1.1. The model for a matrix-like two-dimensional observation sequence will therefore be composed of different (horizontal) one-dimensional models (one for each possible type of line) connected by a global (vertical) model representing relationships between lines. For this reason, models of this type are referred to as *pseudo*-2D models (P2D-model). Figure 1.2 shows an instance of a P2DFSM which is an extension of the simple 1D-model shown in Figure 1.1.

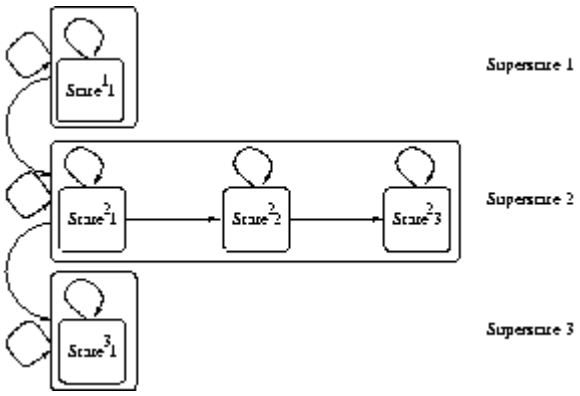

Figure 1.2: A Pseudo-2D FSM.

The notation is extended as follows (see also [6]). The sequence of lines is modelled by an upper level FSM Λ composed of N super-states, each representing a 1D model λ^i. Each 1D model λ^i is composed of N^i states s^i_j. The corresponding superscript (^{i}) is added to every subsequent 1D-model parameter to specify to which model it belongs. Transition probabilities between super-states are given by the values of a_{ij} and transition probabilities between states within the 1D model λ^i are given by the values of a^i_{jk}. Similarly, initial super-state and state probabilities are given by π_i and π^i_j, respectively. Finally, the output probability at state s^i_j is denoted $b^i_j(.)$. Super-state output probability values (*i.e.*, the probability of a line O_y to be generated at a super-state λ^i) are not calculated explicitly since they are simply given by the values of $P[O_y|\lambda^i]$, as calculated in Section 1.1.

1.2.2 Developments

Similarly to one-dimensional modelling, this procedure searches for the most likely state sequence Q^* which corresponds to the given observation sequence O. Two embedded dynamic programming procedures are used. A first step calculates for each line of observations O_y and every possible super-state (*i.e.*, 1DFSM) λ^i the most likely state sequence $q_{1y}, \ldots, q_{X_y}$ through which O_y has been generated. The probability along each possible state sequence is stored using $\delta^i_{xy}(j)$ where,

$$\delta^i_{xy}(j) = \max_{q_{1y},\cdots,q_{x-1y}} P[q_{1y}, \ldots, q_{x-1y}, q_{xy} = s^i_j, o_{1y}, \ldots, o_{xy}|\lambda^i]$$

The resulting optimal moves are stored in the array $\phi^i_{xy}(j)$ so that backtracking can be operated to retrieve this best path. This procedure is as follows.

- $\delta^i_{1y}(j) = \pi^i_j b^i_j(o_{1y})$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y$
- $\delta^i_{xy}(j) = \max\{\delta^i_{x-1y}(l)a^i_{lj}\ ;\ 1 \leq l \leq N^i\} \times b^i_j(o_{xy})$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y, 1 < x \leq X$
- $\phi^i_{xy}(j) = \arg\max\{\delta^i_{x-1y}(l)a^i_{lj}\ ;\ 1 \leq l \leq N^i\}$,
  $1 \leq j \leq N^i, 1 \leq i \leq N, 1 \leq y \leq Y, 1 < x \leq X$

This allows for computing the value of $b_i(O_y) = P[O_y|Q_y = \lambda^i]$ as

$$b_i(O_y) = \max_{1 \leq j \leq N^i} \delta_{X_y}(j)$$

and to obtain the last state in each 1D optimal path using λ^i. This state is stored in $\chi^i(O_y)$ as

$$\chi^i(O_y) = \arg \max_{1 \leq j \leq N^i} \delta_{X_y}(j).$$

A second step finally determines the most likely sequence of super-states $Q_1, \ldots, Q_Y$ corresponding to the sequence of lines of observations $O_1, \ldots, O_Y$ (this corresponds to associating a super-state per line of observations). For each line O_y,

$$\delta_y(i) = \max_{Q_1, \ldots, Q_{y-1}} P[Q_1, \ldots, Q_{y-1}, Q_y = \lambda^i, O_1, \ldots, O_y|\Lambda].$$

This second procedure is as follows.

- $\delta_1(i) = b_i(O_1)$, $1 \leq i \leq N$.
- $\delta_y(i) = \max\{\delta_{y-1}(j)a_{ji} \ ; \ 1 \leq j \leq N\} \times b_i(O_y)$,
  $1 \leq i \leq N, 2 \leq y \leq Y$.
- $\Phi_y(i) = \arg\max\{\delta_{y-1}(j)a_{ji} \ ; \ 1 \leq j \leq N\}$,
  $1 \leq i \leq N, 2 \leq y \leq Y$.

Then,

$$P[O|\Lambda] = \max_{1 \leq i \leq N} \delta_Y(i).$$

The best state sequence backtracking is then done as follows.

- $Q_Y^* = \arg\max\{\delta_Y(i) \ ; \ 1 \leq i \leq N\}$
- For $y = Y - 1, \ldots, 1$:
  - $Q_y^* = \Phi_{y+1}(Q_{y+1}^*)$
  - $q_{Xy}^* = \chi^{Q_y^*}(O_y)$
  - For $x = X - 1, \ldots, 1$:
    $q_{xy}^* = \phi_{xy}^{Q_y^*}(q_{x+1y}^*)$

1.3 A novel 2D Viterbi procedure

In this section, the aim is to develop a dynamic programming procedure that allows for handling 2D connectivity while preserving efficiency in computation and storage. Because of the true 2D structure of data, the concept of 1D FSM has to be reviewed completely.

1.3.1 Principle

The main idea behind this procedure is to take into account the real 2D structure of the image. Causality is defined between pixels and the possible state transitions follow the scheme

$$\begin{array}{ccc} & & o_{xy-1} \\ & & \downarrow \\ o_{x-1y} & \rightarrow & o_{xy} \end{array} \tag{1.2}$$

In other words, based on the idea of transition probabilities ($a_{ij} = P[q_t = s_j|q_{t-1} = s_i]$), we aim to extend the transition probabilities as

$$a_{ijk} = P[q_{xy} = s_k|q_{xy-1} = s_i, q_{x-1y} = s_j].$$

However, as stated in [4], developing a transition graph using this 2D relationship would lead to unmanageable storage and complexity. Moreover, assuming that it is possible to store for each pixel, its best path within the trellis, backtracking would not be straightforward. For example, given x and y, at pixel $p = (x, y)$, the most likely state $q^*_{x,y}$ derives from the states at pixels $p_1 = (x, y-1)$ and $p_2 = (x-1, y)$. Similarly, at pixel $p' = (x-1, y+1)$, the most likely state q^*_{x-1y+1} will be given by the states at pixels $p'_1 = (x-1, y)$ and $p'_2 = (x-2, y+1)$. In that case, the best paths from p and p' "intersect" at $p'_1 = p_2$ and should therefore lead to one unique value. Nothing guarantees such uniqueness, so that a choice will be done, according to some criterion. Rather than pursuing in that direction, we develop the following simplification, in order to avoid contradictions and keep the memory and computational loads reasonable.

1.3.2 Developments

Starting from the origin o_{11}, we detail the emission of the block of pixels $O_{xy} = \{o_{uv}\ ,\ 1 \leq u \leq x\ ,\ 1 \leq v \leq y\}$ corresponding to a state block $Q_{xy} = \{q_{uv}\ ,\ 1 \leq u \leq x\ ,\ 1 \leq v \leq y\}$. Let us define $V_{xy}(k)$ as the maximum probability of a state block Q_{xy} with $q_{xy} = s_k$ corresponding to the emission of pixels in O_{xy}. Formally,

$$V_{xy}(k) = \max_{i,j \leq N}[V_{2D}(x, y, i, j)a_{ijk}]b_k(o_{xy}),$$

where $V_{2D}(x, y, i, j)$ is the maximum probability over all state blocks $Q_{xy} - \{q_{xy}\}$ with $q_{xy-1} = s_i$ and $q_{x-1y} = s_j$ corresponding to the emission of pixels in $O_{xy} - \{o_{xy}\}$. In other words, the state block Q^*_{xy} which maximises this probability is constructed in two major steps. The first step represents the construction of the state block $Q^*_{xy} - \{q_{xy}\}$ through the definition of $V_{2D}(x, y, i, j)$ and the second step (a_{ijk}) represents the two-dimensional transition from states $q^*_{xy-1} = s_i$ and $q^*_{x-1y} = s_j$ to state $q^*_{xy} = s_k$ (and finally the emission of o_{xy}).

The values of a_{ijk} will be given as parameters of our model. The characterisation of $V_{2D}(x, y, i, j)$ requires some definitions. Given that we study state and pixel values at position $p = (x, y)$, we introduce the following notation for simplifying the subsequent equations:

- $p = (x, y)$, $q = q_{xy}$, $o = o_{xy}$,
- $p' = (x-1y-1)$, $q' = q_{x-1,y-1}$, $o' = o_{x-1y-1}$,
- $p_1 = (x, y-1)$, $q_1 = q_{xy-1}$, $o_1 = o_{xy-1}$, $Q_1 = \{q_{uv}\ ,\ 1 \leq u \leq x\ ,\ 1 \leq v \leq y-1\}$,
- $p_2 = (x-1, y)$, $q_2 = q_{x-1y}$, $o_2 = o_{x-1y}$, $Q_2 = \{q_{uv}\ ,\ 1 \leq u \leq x-1\ ,\ 1 \leq v \leq y\}$,

By definition, $V_{2D}(x, y, i, j)$ can be formalised as

$$V_{2D}(x, y, i, j) = \max_{l \leq N, R_{p'}, C_{p'}}[V_A(p', l, R_{p'}, C_{p'})T_{2D}(q_1 = s_i | p', l, R_{p'}, C_{p'})T_{2D}(q_2 = s_j | p', l, R_{p'}, C_{p'})], \tag{1.3}$$

where (see Figure 1.3),

- $R_{p'}$ is one state configuration at positions $\{(u, y)\ ;\ 1 \leq u < x-1\}$ (*i.e.*, along the row containing $p' = (x-1, y-1)$, until the position $(x-2, y-1)$).
- $C_{p'}$ is one state configuration at positions $\{(x, v)\ ;\ 1 \leq v < y-1\}$ (*i.e.*, along the column containing $p' = (x-1, y-1)$, until the position $(x-1, y-2)$).
- $V_A(p', l, R_{p'}, C_{p'})$ is the maximum probability over all state blocks Q_{x-1y-1} corresponding to the emission of O_{x-1y-1} with state s_l at position $p' = (x-1, y-1)$ and containing the configurations of states $R_{p'}$ and $C_{p'}$ on its borders.
- $T_{2D}(q_1 = s_i | p', l, R_{p'}, C_{p'})$ is the maximum probability of over all state blocks $Q_1 = Q_{xy-1}$ corresponding to O_{xy-1} with $q_{xy-1} = s_i$ given the values of s_l at position p', and given that $Q_1 = Q_{xy-1}$ should comprise the state configurations $R_{p'}$ and $C_{p'}$.

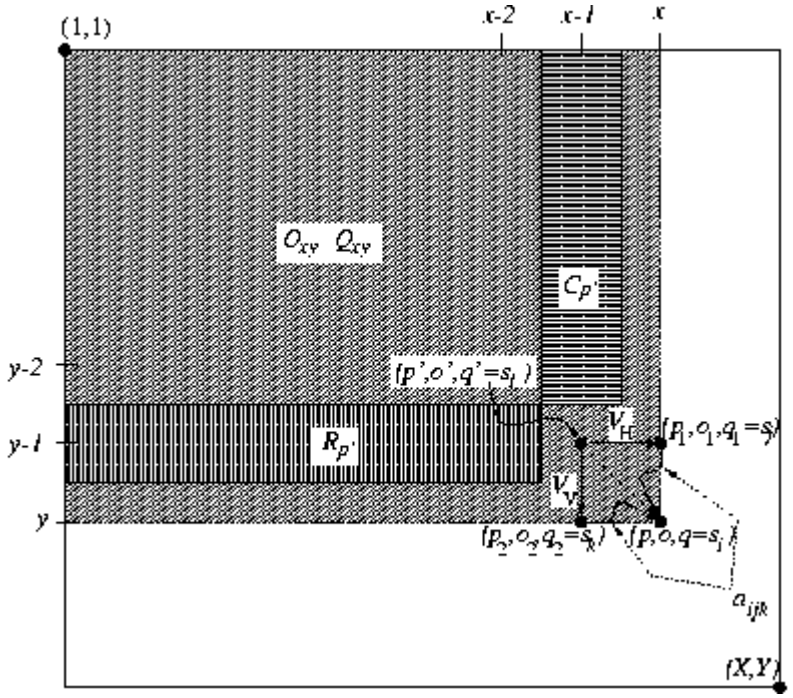

Figure 1.3: Notation for the 2D Viterbi procedure.

The expression of $V_{2D}(x,y,i,j)$ therefore organises the construction of $Q^*_{xy} - \{q_{xy}\}$ into classes of state blocks with respect to row and column configurations they contain (*i.e.*, w.r.t. the values of $R_{p'}$ and $C_{p'}$). The problem is now to keep track of the combinations $(q', R_{p'}, C_{p'})$ leading to maxima of $V_{2D}(x,y,i,j)$. As it is expressed above, Equation (1.3) is untraceable since it requires to store all possible configurations of q', $R_{p'}$ and $C_{p'}$. We therefore suggest the following approximation:

$$V_{2D}(x,y,i,j) \simeq \max_{k \leq N} \left(\left[\max_{R_{p'}, C_{p'}} V_A(p', k, R_{p'}, C_{p'}) \right] \times \left[\max_{R_{p'}, C_{p'}} T_{2D}(q_1 = s_i | p', k, R_{p'}, C_{p'}) \right] \left[\max_{R_{p'}, C_{p'}} T_{2D}(q_2 = s_j | p', k, R_{p'}, C_{p'}) \right] \right),$$

Let us define $V_H(q_1 = s_i | q' = s_k)$ as the maximum probability over all state blocks Q_1 containing the state $q' = s_k$ at position $p' = (x-1, y-1)$ and the state $q_1 = s_i$ at position $p_1 = (x, y-1)$, while emitting O_{xy-1}. In particular,

$$V_H(q_1 = s_i | q' = s_k) = \max_{R_{p'}, C_{p'}} [V_A(p', k, R_{p'}, C_{p'}) \times T_{2D}(q_1 = s_i | p', k, R_{p'}, C_{p'})].$$

Another approximation leads to,

$$V_H(q_1 = s_i | q' = s_k) \simeq \max_{R_{p'}, C_{p'}} V_A(p', k, R_{p'}, C_{p'}) \times \max_{R_{p'}, C_{p'}} T_{2D}(q_1 = s_i | p', k, R_{p'}, C_{p'}). \quad (1.4)$$

Now, by definition,

$$\max_{R_{p'}, C_{p'}} V_A(p', k, R_{p'}, C_{p'}) = V_{p'}(k) = V_{x-1\,y-1}(k).$$

so that,

$$\max_{R_{p'}, C_{p'}} T_{2D}(q_1 = s_i | p', k, R_{p'}, C_{p'}) \simeq \frac{V_H(q_1 = s_i | q' = s_k)}{V_{p'}(k)}.$$

When detailing the case of $p_2 = (x-1, y)$, we define in the same manner $V_V(q_2 = s_i | q' = s_k)$ as the maximum probability over all state blocks Q_2 containing the state $q' = s_k$ at position $p' = (x-1, y-1)$ and the state $q_2 = s_i$ at position $p_1 = (x-1, y)$, while emitting $O_{x-1\,y}$. In this

context,

$$\max_{R_{p'},C_{p'}} T_{2\mathrm{D}}(q_2 = s_i | p', k, R_{p'}, C_{p'}) \simeq \frac{V_{\mathrm{V}}(q_2 = s_i | q' = s_k)}{V_{p'}(k)}$$

We therefore obtain,

$$V_{2\mathrm{D}}(x,y,i,j) = \max_{k \leq N} \left(\frac{V_{\mathrm{H}}(q_1 = s_i | q' = s_k) V_{\mathrm{V}}(q_2 = s_j | q' = s_k)}{V_{p'}(k)} \right).$$

That is,

$$V_{2\mathrm{D}}(x,y,i,j) = \max_{k \leq N} \left(\frac{V_{\mathrm{H}}(q_{xy-1} = s_i | q_{x-1y-1} = s_k) V_{\mathrm{V}}(q_{x-1y} = s_j | q_{x-1y-1} = s_k)}{V_{x-1y-1}(k)} \right).$$

Finally, combining all equations, we obtain the 2D Viterbi recursive formula,

$$V_{xy}(i) = \max_{j,k,l \leq N} \left[\frac{V_{\mathrm{H}}(q_1 = s_j | q' = s_l) V_{\mathrm{V}}(q_2 = s_k | q' = s_l)}{V_{x-1y-1}(l)} P[q_{xy} = s_i | q_{xy-1} = s_j, q_{x-1y} = s_k] \right] P[o_{xy} | q_{xy} = s_i],$$

or,

$$V_{xy}(i) = \max_{j,k,l \leq N} \left[\frac{V_{\mathrm{H}}(q_1 = s_j | q' = s_l) V_{\mathrm{V}}(q_2 = s_k | q' = s_l)}{V_{x-1y-1}(l)} a_{ijk} \right] b_i(o_{xy}),$$

which can equivalently be written as,

$$V_{xy}(i) = \max_{j,k \leq N} [\Delta_{xy}(i,j,k) a_{ijk}] \, b_i(o_{xy}), \tag{1.5}$$

where,

$$\Delta_{xy}(i,j,k) = \max_{l \leq N} \frac{V_{\mathrm{H}}(q_{xy-1} = s_j | q_{x-1y-1} = s_l) V_{\mathrm{V}}(q_{x-1y} = s_k | q_{x-1y-1} = s_l)}{V_{x-1y-1}(l)}.$$

Equation (1.5) clearly shows the idea of an extension from the classic 1D-Viterbi procedure (see Equation (1.1)). It fully uses the principle of a 2D transition from states at positions $p_1 = (x, y-1)$ and $p_2 = (x-1, y)$ to a state at position $p = (x, y)$ (see Figure 1.3). However, in this form, the retrieval of the most likely state sequence Q^* associated to O would be unmanageable since it requires the storage of the values of state indices j, k and l at all positions p_1, p_2 and p', respectively that lead to the most likely state s_i at p. We therefore rewrite these equations as,

$$V_{xy}(i) = \max_{l \leq N} \left[\max_{j,k \leq N} \Psi_{xy}(i,j,k,l) \right] b_i(o_{xy}), \tag{1.6}$$

where,

$$\Psi_{xy}(i,j,k,l) = \frac{V_{\mathrm{H}}(q_{xy-1} = s_j | q_{x-1y-1} = s_l) V_{\mathrm{V}}(q_{x-1y} = s_k | q_{x-1y-1} = s_l)}{V_{x-1y-1}(l)} a_{ijk}, \tag{1.7}$$

These equations now create a link between the resulting most likely state at position $p' = (x-1, y-1)$ and that at position $p = (x, y)$ via the enumeration of all combination of states at positions $p_1 = (x, y-1)$ and $p_2 = (x-1, y)$. Equations (1.6) and (1.7) will form the core of the procedure we propose for retrieving the most likely state sequence from a given sequence of observation O.

1.3.3 2D Viterbi procedure

We now summarise the results obtained earlier and detail the practical implementation of our 2D Viterbi procedure. Because of the two-dimensional aspect of our procedure, storage and computation are clearly increased when compared to that of the 1D- (and P2D-) procedures. Section 1.3.4 analyses the approximations made and the complexity of our new procedure.

A model λ is represented by the following parameters:

- N, the number of states in the model.
- $\mathcal{V}$, the vocabulary (*i.e.*, the set of all possible values of an observation). $\mathcal{V}$ can possibly be of infinite size.
- $A = \{a_{ijk} = P[q_{xy} = s_i | q_{xy-1} = s_j, q_{x-1y} = s_k]\ ;\ 1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq k \leq N\}$, the 2D-transition probability values.
- $B = \{b_i(v) = P[o_{xy} = v | q_{xy} = s_i]\ ;\ 1 \leq i \leq N\ \forall v \in \mathcal{V}\}$, the set of state output probability values.
- $\Pi = \{\pi_i = P[q_{xy} = s_i | x = 1 \text{ or } y = 1]\ ,\ 1 \leq i \leq N\}$, the set of initial state probability which determine the (upper and left) border constraints.

The following intermediate variables are calculated during the forward process,

- $V_{xy}(i)$ is the maximum probability of obtaining a state block Q_{xy} containing the state $q = s_i$ at position $p = (x, y)$, while emitting O_{xy},
- $V_{\mathrm{H}}(q_{xy-1} = s_j | q_{x-1y-1} = s_l)$ is the maximum probability of obtaining a state block Q_1 containing the state $q' = s_l$ at position $p' = (x-1, y-1)$ and the state $q_1 = s_j$ at position $p_1 = (x, y-1)$, while emitting O_{xy-1},
- $V_{\mathrm{V}}(q_{x-1y} = s_k | q_{x-1y-1} = s_l)$ is the maximum probability of obtaining a state block Q_2 containing the state $q' = s_l$ at position $p' = (x-1, y-1)$ and the state $q_2 = s_k$ at position $p_2 = (x-1, y)$, while emitting O_{x-1y},
- $\Psi_{xy}(i, j, k, l)$ is the probability of obtaining one state block Q containing the states s_i, s_j, s_k and s_l at positions $p = (x, y)$, $p_1 = (x, y-1)$, $p_2 = (x-1, y)$ and $p' = (x-1, y-1)$, respectively, while emitting $O_{xy} - \{o_{xy}\}$.

Similarly to the 1D Viterbi procedure (see Section 1.1), the values of V, V_{H}, and V_{V} are defined recursively and will be calculated online during the forward pass of our procedure. The retrieval of the best state sequence Q^* encountered while emitting O will be possible via the storage of states which lead to maximal values during the forward pass. By definition, we first have,

$$P[O|\Lambda] = \max_i V_{XY}(i). \tag{1.8}$$

The backtracking process will therefore be initiated by determining the most likely final state, characterised by,

$$q^*_{XY} = s_{i^*} \text{ where } i^* = \arg\max_i V_{XY}(i).$$

The three surrounding states q_{XY-1}, q_{X-1Y} and q_{x-1Y-1} leading to each possible value s_i of q_{XY} therefore need to be stored during the forward process. In other words, we will fill the array,

$$\Omega_{XY}(i) = (j^*, k^*, l^*) = \arg\max_{j,k,l}[\Psi_{XY}(i, j, k, l)]\ ;\ 1 \leq i \leq N.$$

Then, in order to re-propagate the information along the lower border, we need to store the state $(q_{x-1,Y} = s_k)$ corresponding to the best probability value for the states $(q_{xY} = s_i, q_{xY-1} = s_j)$. In other words, we will fill the array

$$\Theta_{xY}(i, j) = (k^*) = \arg\max_{k,l}[\Psi_{xY}(i, j, k, l)]\ ;\ 1 \leq i, j \leq N,\ 2 \leq x \leq X-1.$$

Similarly, in order to re-propagate the information along the right border, we need to store the state $(q_{X,y-1} = s_j)$ corresponding to the best probability value for the states $(q_{xY} = s_i, q_{X-1y} = s_k)$. In other words, we will fill the array

$$\Gamma_{Xy}(i, k) = (j^*) = \arg\max_{j,l}[\Psi_{Xy}(i, j, k, l)]\ ;\ 1 \leq i, k \leq N,\ 2 \leq y \leq Y-1.$$

Finally, for all pixels $p(x, y)$ but that on the bottom and right borders, the backtracking is done via the storage of the state $q' = q_{x-1y-1} = s_l$ at position $p' = (x-1, y-1)$ corresponding to the

best probability of having $q_{xy} = s_i$, $q_{xy-1} = s_j$ and $q_{x-1y} = s_k$. In others words, we will fill the array

$$\Phi_{xy}(i,j,k) = l^* = \arg\max_l [\Psi_{xy}(i,j,k,l)] \ ; \ 1 \leq i \leq N, \ 2 \leq x \leq X-1, \ 2 \leq y \leq Y-1,$$

during the forward pass.

The complete 2D Viterbi procedure can now be described as follows.

Forward pass:

Initialisation (upper and left borders):
For all (x,y) such that $x = 1$ or $y = 1$,

- $V_{xy}(i) = \pi_i b_i(o_{xy})$; $1 \leq i \leq N$,
- $V_{\mathrm{H}}(q_{xy} = s_i | q_{x-1y} = s_k) = \pi_i$; $1 \leq i \leq N$, $1 \leq k \leq N$,
- $V_{\mathrm{V}}(q_{xy} = s_i | q_{xy-1} = s_k) = \pi_i$; $1 \leq i \leq N$, $1 \leq k \leq N$.

Main loop (all pixels in raster-scan order):
For all $2 \leq y \leq Y$, $2 \leq x \leq X$,

- $V_{xy}(i) = \max_l [\max_{j,k} \Psi_{xy}(i,j,k,l)] b_i(o_{xy})$; $i = 1, \ldots, N$,
- $V_{\mathrm{H}}(q_{xy} = s_i | q_{x-1y} = s_k) = \max_{j,l} [\Psi_{xy}(i,j,k,l)] b_i(o_{xy})$; $i,k = 1, \ldots, N$,
- $V_{\mathrm{V}}(q_{xy} = s_i | q_{xy-1} = s_j) = \max_{k,l} [\Psi_{xy}(i,j,k,l)] b_i(o_{xy})$; $i,j = 1, \ldots, N$.

Storage for best state sequence retrieval

- $\Phi_{xy}(i) = \arg\max_l [\max_{j,k} \Psi_{xy}(i,j,k,l)]$;
  $i = 1, \ldots, N$, $x = 2, \ldots, X-1$, $y = 2, \ldots, Y-1$,
- $\Gamma_{Xy}(i,k) = \arg\max_j [\max_l \Psi_{xy}(i,j,k,l)]$;
  $i,k = 1, \ldots, N$, $y = 2, \ldots, Y-1$,
- $\Theta_{xY}(i,j) = \arg\max_k [\max_l \Psi_{xy}(i,j,k,l)]$;
  $i,j = 1, \ldots, N$, $x = 2, \ldots, X-1$,
- $\Omega_{XY}(i,j,k) = \arg\max_l [\Psi_{xy}(i,j,k,l)]$.

Backtracking:

The most likely 2D state sequence $Q^* = \{q^*_{xy} \ ; \ 1 \leq x \leq X, \ 1 \leq y \leq Y\}$ can now be retrieved as follows[2].

- $P[O|\Lambda] = \max_i V_{XY}(i)$,
- $q^*_{XY} = \arg\max_i V_{XY}(i)$,
- $(q^*_{XY-1}, q^*_{X-1Y}, q^*_{X-1Y-1}) = \Omega_{XY}(q^*_{XY})$,
- $(q^*_{xY}) = \Theta_{xY}(q^*_{x+1Y}, q^*_{x+1Y-1})$; $x = X-2, \ldots, 1$,
- $(q^*_{Xy}) = \Gamma_{Xy}(q^*_{Xy+1}, q^*_{X-1y+1})$; $y = Y-2, \ldots, 1$,
- $q^*_{xy} = \Psi(q^*_{x+1y+1})$; $y = Y-1, \ldots, 1$, $x = X-1, \ldots, 1$.

[2] Here, we extend the notation so that "$q_{xy} = i$" is equivalent to "$q_{xy} = s_i$".

1.3.4 Analysis

During the theoretical developments, approximations were made in order to simplify the procedure both in memory requirement and complexity. This is studied in depth in this section.

Approximations

The first approximation is made from Equation (1.3). It decouples the best paths leading to q_1^*, q_2^* and q'^*, respectively. Instead of considering the construction of the best state sequence $Q_1^* \cup Q_2^*$ as that of one unique state sequence corresponding to the emission of $O_1 \cup O_2$, it builds them independently with still keeping the constraint of a common state $q^* = s_k$. This approximation therefore reduces the dependency of the state at position (x, y) to only the values of states at q_{x-1y-1}, q_{x-1y} and q_{xy-1}.

This in turn also to reduce dramatically the amount of information needed to backtrack the best path. We now only need to store the values of q_{x-1y-1}, q_{x-1y} and q_{xy-1} leading to the state of highest probability at q_{xy}. The backtracking will therefore be done in a trellis such as that schematically shown in Figure 1.4 and whose (x, y)-projection is presented in Figure 1.5.

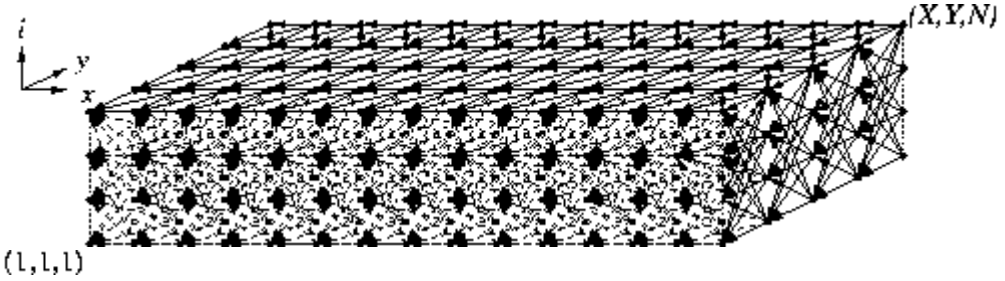

Figure 1.4: Path trellis used by the 2D Viterbi procedure.

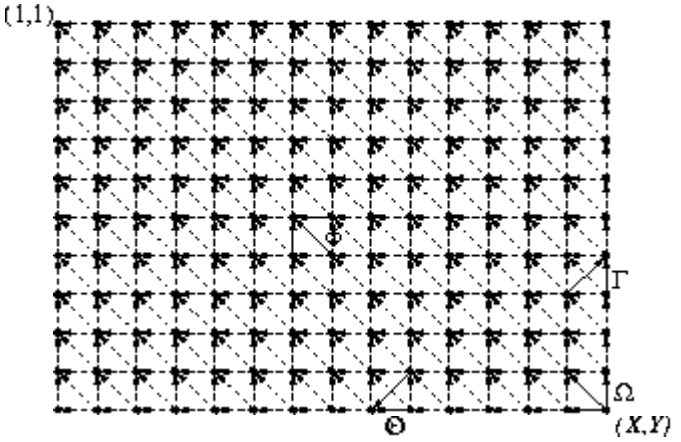

Figure 1.5: Spatial (x, y)-projection of the path trellis used by the 2D Viterbi procedure. Instances of array values are shown.

The second approximation is similar to the previous in that it reduces the dependency of state $q_{x,y-1}$ to the values of q_{x-1y-1}, given a certain state configuration Q_{x-1y-1} (see Equation (1.4)). It should be noted that in a complete and exact procedure, given Q_{x-1y-1}, all states q_{xv} $(v = 1, \ldots, y-1)$ should be determined in an inter-dependent fashion, thus imposing the need for storing all such configurations.

Memory load

An upper bound for the memory requirements of the procedure is straightforward to estimate.

Variable	Size
Ω	$3 \times N$
Θ	$N \times N \times (X-1)$
Γ	$N \times N \times (Y-1)$
Φ	$(X-1) \times (Y-1) \times N \times N \times N$
V	$X \times Y \times N$
V_{H}	$X \times Y \times N \times N$
V_{V}	$X \times Y \times N \times N$
Total	$\mathcal{O}(X \times Y \times N^3)$

Storage requirements can be divided in $\mathcal{O}(X \times Y \times N^2)$ for the forward pass and $\mathcal{O}(X \times Y \times N^3)$ for storing the necessary information for backtracking. In other words, if no backtracking is needed (only $P[O|\Lambda]$ is to be estimated) then only $\mathcal{O}(X \times Y \times N^2)$ values need to be stored for calculation.

Backtracking information can be significantly reduced by storing only values corresponding to state configuration leading to global probability values above a certain threshold. In this manner, since we are eventually just interested is the most likely path, the least likely paths, emerging from unlikely combinations of states are not stored.

CPU load

For filling the backtracking arrays and determining the value of $P[O|\Lambda]$, a number of calculations following $\mathcal{O}(X \times Y \times N^4)$ is imposed by Equation (1.7). Again, this number of calculations may be reduced using fast estimation of unlikely paths. Such fast estimation is done for each state configuration (s_i, s_j, s_k, s_l) corresponding to the location (x, y) (see Figure 1.3) by first examining $b_i(o_{xy})$ to see whether observation o_{xy} can be produced by state s_i. Then the value of a_{ijk} will indicate whether this 2D transition is permitted within the model description. Then checking the values of $V_{\mathrm{H}}(q_{xy-1} = s_j | q_{x-1y-1} = s_l)$, $V_{\mathrm{V}}(q_{x-1y} = s_k | q_{x-1y-1} = s_l)$ and $V_{x-1y-1}(l)$ will indicate whether the most likely paths leading to the particular state configuration (s_j, s_k, s_l) are possible or otherwise. If all these tests are positive then calculations are made and the backtracking information stored.

Experimental test

These complexities are tested experimentally in the following way. First an image of size 36×36 is taken and a model comprising N states in the form of a regular grid is considered. The times needed for the computation of the state segmentation are reported as N augments. This is illustrated by

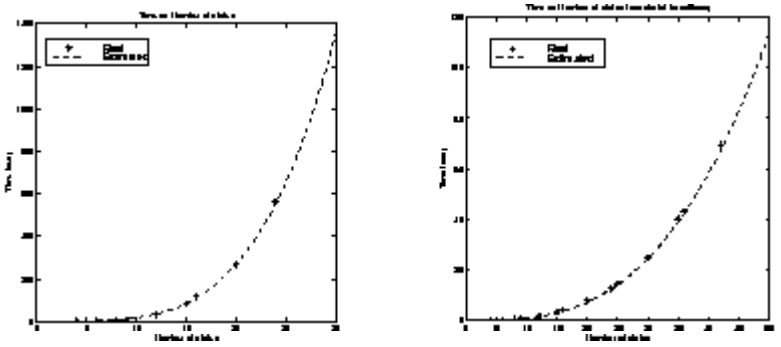

Figure 1.6: Time versus number of states for the segmentation procedure

Figure 1.6 where the dependence is shown to follow $T \simeq \mathcal{O}(N^4)$ ($T \simeq \frac{N^4}{1100}$) when no simplification is made (*i.e.*, when state transitions are not constrained and therefore all paths are investigated). When constraints are taken into consideration for simplifying the procedure and reducing the complexity, the dependency is of order $T \simeq \mathcal{O}(N^3)$ ($T \simeq \frac{N^3}{600}$). Figure 1.7 highlights the reduction in time when constraints are taken into consideration.

The state number is now fixed to $N = 16$ and the image size $X = Y$ is increased. Figure 1.8 reports the results and shows that the dependence is of order $T \simeq \mathcal{O}(XY)$ ($T \simeq \frac{XY}{260}$).

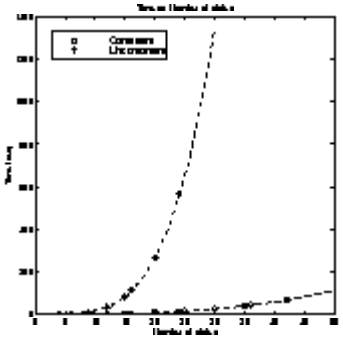

Figure 1.7: Time reduction when using the constrained segmentation procedure

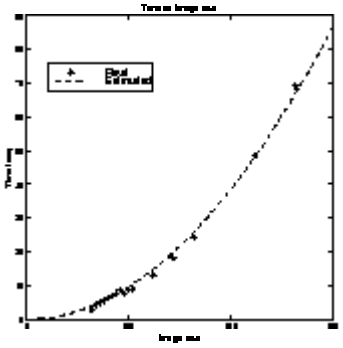

Figure 1.8: Time versus image size

During these experiments, the memory allocated for the procedures have been recorded and results are reported in Figure 1.9.

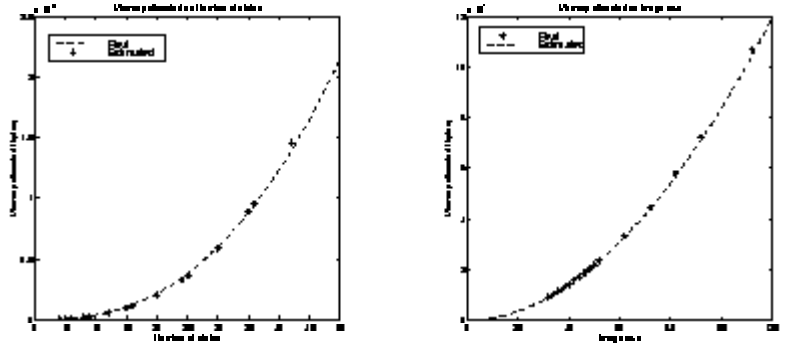

Figure 1.9: Memory allocated versus number of states and image size

It should be noted that these figures correspond to the maximum *simultaneous* memory needed by the procedure (as opposed to the *total throughout*). From this data, it was found that the memory load was following $\mathcal{O}(N^{2.5})$ for a fixed image size and $\mathcal{O}(XY)$ for a fixed number of states. These figures correspond the expected complexities.

1.4 Chaining the 2D models

The basic idea in chaining the models into a upper-level (meta) structure is to add null states to the respective models so that transitions between models are made through these null states (see Figure 1.10). This is a technique commonly used in speech processing where each phoneme model starts and ends by such a null state and grammars of models can be designed by defining the transition topologies between these null states.

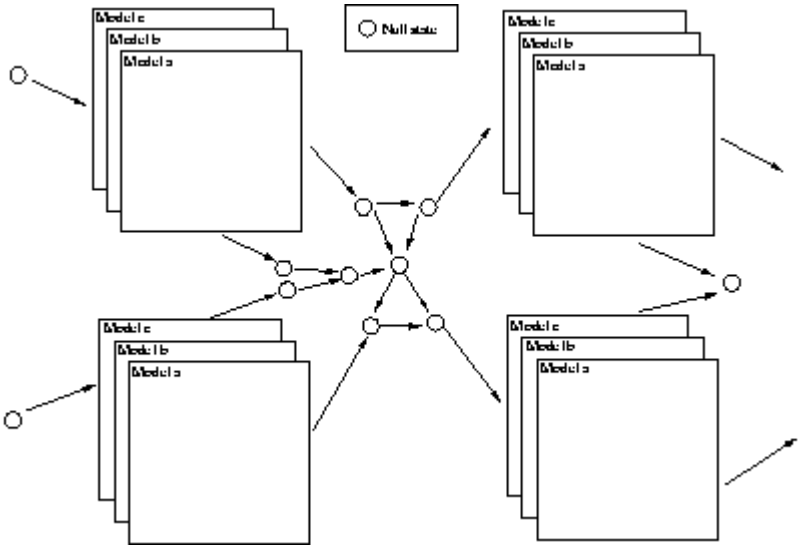

Figure 1.10: Model chaining

A null state is a state which does not emit any observation. It should therefore be treated differently in the backtracking procedure since an observation will never be associated to a null state. Rather, null states compose a sub path in the trellis, completely contained into one single position (*i.e.*, pixel).

1.4.1 Notation

Using the same notation as before, and adding a superscript index for the 2D-model in question (*e.g.*, s_i^a for state i of model a), we consider the cases where the transition is made between null and/or emitting states. Note that, in the following developments, the fact of associating each state to a particular local 2D model is done for the ease of understanding when relating to the previous developments. Moreover, when keeping a state associated with a local model, the backtracking can be done at two levels, the state level and the model level. This superscripting has the inconvenient of complicating the notation, and, more importantly, does not permit the definition of null state associated with no models which would be used in the definition of complex grammars, as it is done in speech processing.

In a first approach we consider that states are associated with local model and then discuss the extension to a generic meta-model construction.

1.4.2 Inner-model non-null state transition

This is the case when three states of a model a yield a state within model a. Typically, calculations are the same as that detailed previously (Figure 1.3, Equations (1.6) and (1.7)) so that

$$V_{xy}(a,i) = \max_{l \leq N^a} \left[\max_{j,k \leq N^a} \Psi_{xy}^a(i,j,k,l) \right] b_i^a(o_{xy}), \tag{1.9}$$

where,

$$\Psi_{xy}^a(i,j,k,l) = \frac{V_{\mathrm{H}}(q_{xy-1} = s_j^a | q_{x-1y-1} = s_l^a) V_{\mathrm{V}}(q_{x-1y} = s_k^a | q_{x-1y-1} = s_l^a)}{V_{x-1y-1}(a,l)} a_{ijk}^a. \tag{1.10}$$

Again backtracking is done by storing the configuration of states $(s_i^a, s_j^a, s_k^a, s_l^a)$ that lead to the highest probability value at a given position (x,y) (see Section 1.3.3).

1.4.3 Transition from a null state

We first present how to "exit" from a null state. Assuming that the probability of being at a null state s_i^a at position (x, y) has been computed, we calculate the probability of moving from that state to another state s_j^b. In our calculation, we assume that null state transitions at a position only arise once the emission of the observation has been made. This ensures that null state transitions happen only after emission at one position. Under this assumption transition to an emitting state will be made at a further position.

In this context, s_j^b is also a null state. Therefore, this transition represents an inner-state transition since no observation is emitted, neither at state s_i^a, nor at state s_j^b. In that case, this is equivalent to splitting the state at position (x, y) into two sub-states (see Figure 1.11).

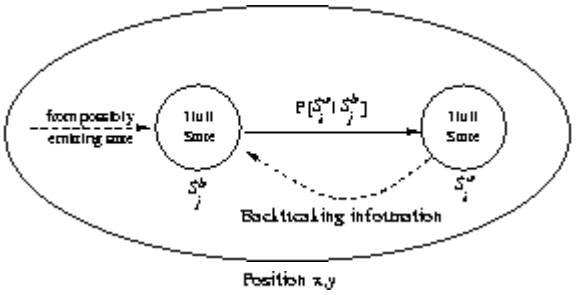

Figure 1.11: Null to null state transition

Therefore, assuming that $V_{xy}(i, a)$ is known, we obtain

$$V_{xy}(j, b) = V_{xy}(i, a) P[s_j^b | s_i^a]$$

The value of $P[s_j^b | s_i^a]$ being taken from the parameters of the model. When generalising, the value associated with s_j^b will represent the maximal possible value for all transition from null states such as s_i^a. As backtracking information the state s_i^a which produced this maximal value is retained.

Assuming now that all state probabilities have been calculated at all positions preceding (x, y) in a row scanning order, calculating the transitions to states at (x, y) start by the calculation of the probability of being in emitting state at (x, y). A transition from a null state is therefore simply treated as a normal 2D-state transition. One major difference is the evaluation of the 2D transition probability value a_{ijk}. Since at (x, y) the local model may change, there is the need for the definition of a value of

$$a_{ijk}^{a,b,c} = P[q_{xy} = s_i^a | q_{x-1y} = s_j^b, q_{xy-1} = s_k^c]$$

and this, in a way, defeats the advantages of keeping each state associated to a local model.

$$V_{xy}(a, i) = \max_{l \leq N^d} \left[\max_{j \leq N^b, k \leq N^c} \Psi_{xy}^{a,b,c,d}(i, j, k, l) \right] b_i^a(o_{xy}), \tag{1.11}$$

where,

$$\Psi_{xy}^{a,b,c,d}(i, j, k, l) = \frac{V_{\mathrm{H}}(q_{xy-1} = s_j^a | q_{x-1y-1} = s_l^d) V_{\mathrm{V}}(q_{x-1y} = s_k^a | q_{x-1y-1} = s_l^d)}{V_{x-1y-1}(d, l)} a_{ijk}^{a,b,c}. \tag{1.12}$$

This formula is simplified by the used of one global model only, as detailed in Section 1.4.5.

1.4.4 Transition to a null state

This step is the basis for chaining models. Adding null states to the models allows for the definition of a model connectivity and therefore a model grammar. Null states are associated to border states so that model can be connected by their borders. Hence, it is natural to think that null states will be divided into resembling East, West, North and South null states.

We detail here how the calculation are made when moving from a emitting state to a null state. Null states have the property that they do not emit any observation (*i.e.*, they are not associated with observation PDF). Following the earlier analysis, null state transition can be thought of as being "in between" emitting state transition. Since transitions to null states happen at one position only after a transition to an emitting state at the same position, we consider the dependency of a null state for backtracking as being the previous emitting state.

In this context, one way to calculate the probability of moving to a null state s_i^a from a given emitting state s_j^b may be calculated as

$$V_{xy}(i,a) = V_{xy}(j,b)P[s_i^a|s_j^b]$$

However, since null state represent borders of a local model, it may be useful to enhance the coherence within such border. For achieving such coherence, one may calculate the dependency of the null state at some position based on states occurring at vertical and horizontal neighbouring positions (see Figure 1.12).

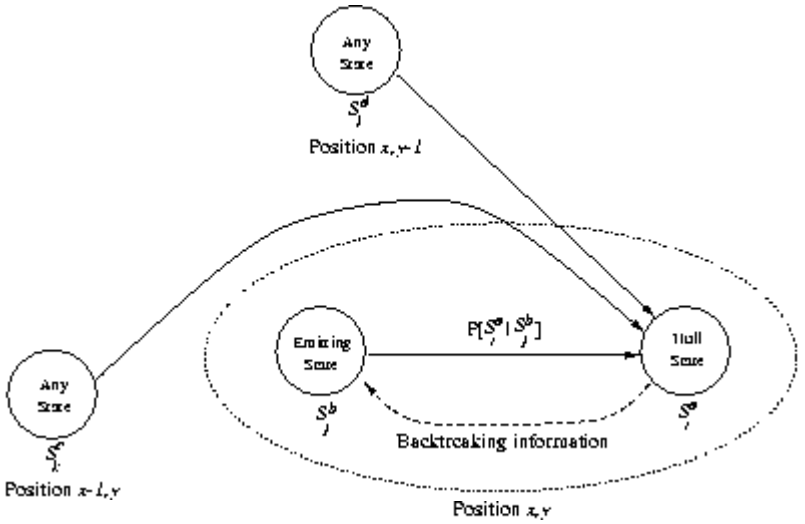

Figure 1.12: Emitting to null state transition

In that context,

$$V_{xy}(i,a) = \max_{j,k,l} \frac{V_{\mathrm{H}}(q_{xy} = s_j^b | q_{x-1y} = s_k^c) V_{\mathrm{V}}(q_{xy} = s_j^b | q_{xy-1} = s_l^d)}{V_{xy}(b,j)} P[s_i^a | s_j^b, s_k^c, s_l^d]$$

1.4.5 Using one unique global model

Using different local models that are trained separately may be a convenient way to represent a global model structure. With the above developments, each local model may be added with null state used as an interface for connecting it to another model. In that respect inter-emitting state transitions may only happen within a local model. Therefore, the Equations (1.11) and (1.12) are simplified as

$$V_{xy}(a,i) = \max_{l \leq N^a} \left[\max_{j,k \leq N^a} \Psi_{xy}^a(i,j,k,l) \right] b_i^a(o_{xy}), \tag{1.13}$$

where,

$$\Psi_{xy}^a(i,j,k,l) = \frac{V_{\mathrm{H}}(q_{xy-1} = s_j^a | q_{x-1y-1} = s_l^a) V_{\mathrm{V}}(q_{x-1y} = s_k^a | q_{x-1y-1} = s_l^a)}{V_{x-1y-1}(a,l)} a_{ijk}^a. \tag{1.14}$$

Thus diminishing the complexity of storage.

An upper transition matrix may be constituted using inner-model transition matrices a_{ijk}^a, added with a matrix describing the null state transition topology and therefore the "grammar" of

the upper-level model.

This way may therefore be seen as exploiting one unique global model while keeping separated local properties for the inter-emitting state transitions.

1.4.6 Null state sequences

The extension of the use of null state for defining a "grammar" for the upper-level model may induce the possibility of null state sequence and therefore null state cycles.

This may be limited when backtracking by either marking the path, as in a shortest path search or by defining an appropriated transition topology associating a sufficient "cost" to null state cycles.

Chapter 2

Applications

We first detail the current implementation of the procedure and give some example applications for the ongoing validation of this procedure.

2.1 Practical implementation

We briefly review the main features of the implementation of the above theoretical procedure. This implementation is done in the most general context and developments are made to handle 2D data arising from both discrete and continuous multidimensional distributions.

One major structure `trellis2D_t` is used to store the model Λ and is composed of a hierarchy of component whose encapsulation is described as follows:

```
typedef struct
{
 char *streamLabel;
 unsigned int globalDim;
 boolean_t full;
 boolean_t discrete;
}streamInfo_t;

typedef struct
{
 streamInfo_t *streamInfo;
 FLOAT *meanVect;
 FLOAT **varMat;
 FLOAT detVarMat;
}PDF_t;

typedef struct
{
 streamInfo_t *streamInfo;
 boolean_t *inState;
 char *label;
 unsigned int nMixture;
 FLOAT *mixCoef;
 PDF_t *PDF;
 char init[10];
}contPDF_t;

typedef struct
{
 streamInfo_t *streamInfo;
 unsigned int nBins;
```

```
 FLOAT minVal;
 FLOAT maxVal;
 FLOAT nVal;
 FLOAT *binVal;
 FLOAT *logBinVal;
}histo_t;

typedef struct
{
 streamInfo_t *streamInfo;
 boolean_t *inState;
 char *label;
 histo_t *histo;
 char init[10];
}discPDF_t;

typedef struct
{
 boolean_t nullState;
 streamInfo_t *streamInfo;
 unsigned int outPDFIndex;
 char *label;
 contPDF_t *contPDF;
 discPDF_t *discPDF;
 char init[10];
}outPDF_t;

typedef struct
{
 boolean_t loaded;
 boolean_t set;
 streamInfo_t streamInfo;
 unsigned int maxMixture;
 unsigned int nState;
 char *label;
 FLOAT *upperStateProb;
 FLOAT *logUpperStateProb;
 FLOAT *leftStateProb;
 FLOAT *logLeftStateProb;
 FLOAT *lowerStateProb;
 FLOAT *logLowerStateProb;
 FLOAT *rightStateProb;
 FLOAT *logRightStateProb;
 FLOAT *nullNStateProb;
 FLOAT *logNullNStateProb;
 FLOAT *nullEStateProb;
 FLOAT *logNullEStateProb;
 FLOAT *nullSStateProb;
 FLOAT *logNullSStateProb;
 FLOAT *nullWStateProb;
 FLOAT *logNullWStateProb;
 FLOAT ***transProb;
 FLOAT ***logTransProb;
 outPDF_t **outPDF;
 unsigned int nCommonOutPDF;
 outPDF_t *commonOutPDF;
 contPDF_t *commonContPDF;
```

```
 discPDF_t *commonDiscPDF;
 char init[10];
}trellis2D_t;
```

For flexibility in the design, each possible output distribution is considered as a separate object and is pointed by structure where needed.

Discrete observation distributions (*i.e.*, cases where the vocabulary V is of finite length) are stored as histograms H_d in each direction and the global output probability value of a D-dimensional observation $o = (o^{(1)} \dots o^{(D)})\top$ is given by

$$b_i(o) = \prod_{d=1}^{D} H_i(o^{(d)}).$$

In other words, all dimension of the observation are assumed to follow independent distributions.

In the case of continuous distribution, depending on whether the flag `streamInfo.discrete` is set or not, the distribution is modelled by a D-dimensional Gaussian mixture with full covariance matrix Σ,

$$b_i(o) = \sum_{j=1}^{m_i} \frac{1}{\sqrt{2\pi|\Sigma_{jk}|}} \exp\left(-\frac{1}{2}(o-\mu_{jk})^\top \Sigma^{-1}(o-\mu_{jk})\right),$$

or diagonal covariance matrix (diag$(\sigma^{(d)})$)

$$b_i(o) = \sum_{j=1}^{m_i} \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi}\sigma_{jk}^{(d)}} \exp\left(\frac{-(o^{(d)}-\mu_{jk}^{(d)})^2}{2\sigma_{jk}^{(d)2}}\right),$$

depending on whether the flag `streamInfo.full` is set or not, respectively.

`null7StateProb` values are used in the model-chaining procedure (`multi1`) to determine the probability from a emitting state to map onto a null state. These are simply calculate by considering the proportion of each state on each border of the training image (when training each T2D model separately).

The tag `nullState` in the structure `outPDF_t` allows for differentiating null and emitting states. The property is therefore carried throughout the process and should be exploited for calculating maximal probabilities at every state.

All parameters needed for initialisation are stored into configuration files and examples of such files are given as Appendix A.

Another structure is used for chaining the models and is as follows:

```
typedef struct
{
 boolean_t loaded;
 boolean_t set;
 int nModel;
 char **fileName;
 int streamDim;
 int maxNState;
 char *label;
 trellis2D_t *T2D;
 FLOAT ******transProb;
 FLOAT ******logTransProb;
 FLOAT **hTransProb;
 FLOAT **logHTransProb;
}model2D_t;
```

A `model2D_t` structure actually simply consists in storing a list of pointers on the independent T2D models and their respective transition probabilities. It is therefore within this structure that all upper level parameters will be stored and carried throughout the procedures.