INFORMys: A Flexible Invoice-Like Form-Reader System

Francesca Cesarini, Marco Gori, *Senior Member, IEEE*,
Simone Marinai, and Giovanni Soda, *Member, IEEE*

Abstract—In this paper, we describe a flexible form-reader system capable of extracting textual information from accounting documents, like invoices and bills of service companies. In this kind of document, the extraction of some information fields cannot take place without having detected the corresponding instruction fields, which are only constrained to range in given domains. We propose modeling the document's layout by means of attributed relational graphs, which turn out to be very effective for form registration, as well as for performing a focussed search for instruction fields. This search is carried out by means of a hybrid model, where proper algorithms, based on morphological operations and connected components, are integrated with connectionist models. Experimental results are given in order to assess the actual performance of the system.

Index Terms—Attributed relational graphs, document analysis and recognition, document registration, invoice processing, location of information fields.

1 Introduction

Document image analysis and recognition have received much attention in the last years. Among different applications, financial document processing [1] and forms processing [2], [3], [4] are becoming very important in practice, since those documents have typically a simple structure when compared with general documents, that like articles, include complex figures and mathematical symbols. Moreover, the automation of the form reading process is an objective of relevant interest, since forms processing is in fact an essential operation in many business organizations.

The overall objective of a form reading system is to obtain a symbolic representation of the data (either printed or handwritten) from a given form. The typical operations involved in these systems are document image acquisition and preprocessing, layout analysis, and data interpretation [1], [3]. Most systems execute these steps sequentially by a bottom-up approach derived from classical image analysis approaches.

The *preprocessing* step usually involves binarization followed by form registration, which is often performed by bottom-up operations [3], [5]. During the layout analysis, important regions which contain the information are located in the form. Throughout the paper, these regions are referred to as information fields. Finally the *data interpretation* step takes place on regions previously extracted and can involve either handwritten or printed character recognition.

- *F. Cesarini, S. Marinai, and G. Soda are with Dipartimento di Sistemi e Informatica, Università di Firenze. Via S. Marta, 3 - 50138 Firenze, Italy. E-mail: (cesarini, simone, giovanni)@mcculloch.ing.unifi.it.*
- *M. Gori is with Dipartimento di Ingegneria dell'Informazione, Università di Siena Via Roma, 56 - 53100 Siena, Italy. E-mail: marco@mcculloch.ing.unifi.it.*

Manuscript received 25 Sept. 1997; revised 4 May 1998. Recommended for acceptance by R. Kasturi.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106853.

In many practical cases, form recognition is simply based on the assumption that the information fields are in fixed positions. In other cases (see e.g., Tang et al. [1]) the information fields are extracted thanks to the detection of lines. Each line is described by its orientation, thickness, and position in the form, and serves as a reference for detecting information fields. Most approaches that hardly rely simply on line detection, however, are quite limited in that they can only deal with table-like forms. Other approaches (see, e.g., [6]) are based on the description of the form layout by means of rectangles instead of lines. This kind of description is obviously more compact but, on the other hand, it is not as much powerful as that based on line detection.

A more flexible way to characterize information fields is that of describing their position with respect to the corresponding instruction field, which describes the content of an information field. In the system proposed in [7], only fields that are inside rectangles (referred to as frames) are taken into account. Two classes of frames are considered: label frames, containing instruction fields, and data frames, containing information fields. The link between a label and the corresponding data frame is given by the spatial relationship between the two rectangles. The system described by Lam and Srihari [8] locates related instruction and information fields by recognizing the text of the instruction field.

The sequence of operations needed to carry out form registration and layout analysis depends on whether or not the form is known in advance [9]. Consequently, model-driven and data-driven approaches can be adopted and, in some cases, these approaches can profitably be integrated [10]. If the form is *known* in advance, that is, if its layout is exactly defined, then the data extraction can naturally be based on a top-down, model-driven process [11]. In these systems, the form registration turns out to be the major problem, as data can directly be extracted once the registration has taken place. In the case of *unknown* forms, a possi-

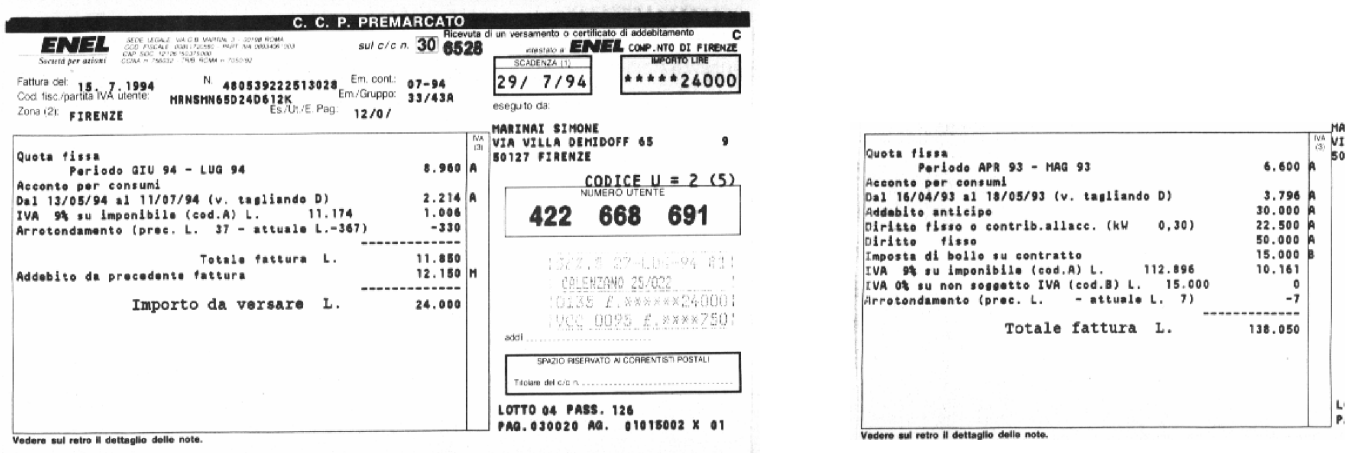


Fig. 1. A typical bill emitted by Italian Electrical Company ENEL. As can be seen on the right side (containing a portion of another bill), the largest box has no fixed structure.

ble approach is that of classifying documents in a fixed number of classes, and then use the model corresponding to each document class to carry out the layout analysis [2], [6]. An intermediate case is that of forms of *known class*, where the layout has not a fixed geometrical structure, but can instead only be inferred on the basis of the instruction fields [4], [7], [8].

1.1 An Overview of INFORMys

INFORMys,[1] the system described in this paper, is designed to deal with *known-class forms* and has been mainly conceived to deal with forms like those issued by service companies for accounting (see e.g., Fig. 1). Like the models proposed in [4] and [12], INFORMys is based on graphs for describing the form layout. However, instead of using hierarchical [4] or nonhierarchical graph [12] with symbolic spatial relationships (e.g., *above, left*), our model is based on nonhierarchical attributed relational graphs (ARG) [13], [14]. ARGs are typically used for image representation and analysis in computer vision. The nodes describe objects or parts of objects, while the arcs describe the mutual relationships between the nodes by means of numerical attributes. In our model, the form layout is described by means of an attribute relational graph referred to as the *form graph*, where the nodes represent lines, instruction fields, information fields, and logos, while the arcs represent the mutual position of the items corresponding to the linked nodes. Concerning numerical arc attributes, instead of using only the distance between objects, as in [14], we consider a vector which connects the barycenters of the objects. By using ARGs, we have an accurate and flexible description of the form class. A bottom-up approach could be used in which one creates a graph describing the incoming unknown form, and subsequently matches the items of that graph with those of the form graph used as a reference to model the accounting forms [4]. However, this would give rise to a very expensive algorithm, and it does not seem to be the best way to deal with documents in which the structure is known.

1. **INFORMys** is an acronym for "flexible **IN**voice-like **FORM**-reader s**ys**tem."

We suggest using a method in which a top-down processing is switched to a bottom-up processing on the basis of the information on the form graph. *Form registration* is performed by using an algorithm based on a hypothesize-and-verify method [15]. The alignment methods used in computer vision to register the position of rigid objects [15], [16] are based on correspondences of data and model features of dimensionality sufficient to compute a complete transformation. Each transformation constitutes a hypothesis about the pose of the object which must be verified. By verification we mean that additional evidence, either supporting or refuting this hypothesis, must be accumulated in the image [15]. In our approach to form registration, the features used to evaluate the form position are stored as nodes of a subgraph of the form graph, which is referred to as the registration graph. The hypothesized transformation is computed by aligning a model feature, e.g., a line node, with a corresponding feature in the incoming form. The verification is carried out by searching those form items that match all the nodes of the registration graph. If the verification fails then a new hypothesis is generated and subsequently verified until the verification succeeds or there are no more hypotheses to test.

Concerning the *layout analysis*, we propose an approach which is based on the location of both logos and instruction fields that, subsequently, make it possible the extraction of the information fields, thanks to our graph-based model. A related approach was proposed in [8], but the instruction fields were limited to keywords. Word recognition has been extensively studied in the literature with different approaches (e.g., see [17], [18]). Methods for word recognition can be classified depending on whether or not the word being recognized is segmented into single characters. The segmentation into characters turns out to be the only pursuable way for recognizing words of a very large dictionary, whereas very effective algorithms can be conceived for the recognition of words of small dictionaries when using the whole word as input to the classifier. In order to deal with highly-noisy forms including also graphical items and to speed up the recognition, we use a connectionist-based model in which both keywords and logos are processed in the same way by simply regarding them as input patterns.

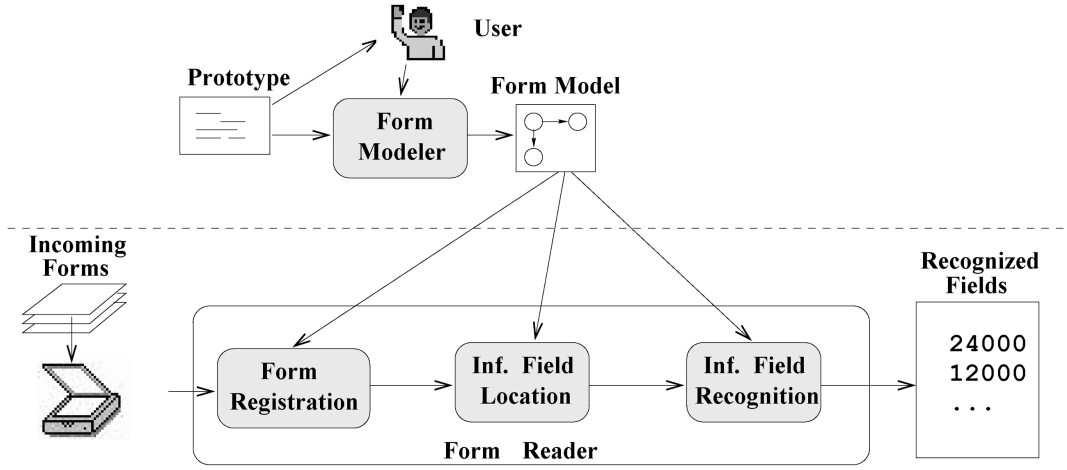


Fig. 2. The basic structure of INFORMys, a flexible form-reader system, where the user can define the layout of its own documents in the class of invoice-like documents.

The keyword recognition accuracy is improved by integrating the hypothesis obtained from the whole word, with an OCR module charged of recognizing the single characters which compose the word. This module is only fired once the connectionist-based model processing the whole keyword gives no enough recognition confidence.

Unlike *instruction fields*, which belong to a finite dictionary, the recognition of *information fields* must necessarily be based on the recognition of the single characters. We used a connectionist-based model for character recognition that is strongly inspired by the work described in [19]. Like most form-reader systems, INFORMys offers a twofold user interface for assisting the user during both form modeling and reading. The *Form Modeler* (*FM*) assists the user building the form graph, while the *Form Reader* (*FR*) implements the recognition engine (see Fig. 2).

INFORMys' user interface is based on X-window[2] and supports interactive functions for defining the form structure. During the construction of the form graph, the operator points to the objects (lines, instruction and information fields, logos) that are relevant for representing the structure of the forms. These objects correspond to the nodes of the form graph, while the arcs express the mutual relationships of different objects. The location of objects and of their mutual relationship is carried out by the user, and the corresponding information is subsequently exploited to create the form graph according to the specifications given in Section 2.

The form-reader module is responsible for supporting the recognition phase. The form reading is a sequential process where the incoming form is first scanned and converted into an electronic image, then aligned by form registration, and, finally, the information fields are located and recognized.

INFORMys deals with documents which contain four different kinds of "objects," namely, lines, logos, instruction fields, and information fields. Lines, logos, and instruction fields are useful for locating the information field and, therefore, throughout this paper, are referred to as *reference objects*. A document class is identified by specifying the possible ways of relating the information to instruction fields and other objects. The class is defined when specifying exactly the objects and their relationships in the documents.

The paper is organized as follows. Section 2 describes in detail the form graph-based model, while the registration algorithm is given in Section 3. Section 4 describes the location of logos, instruction fields, and information fields. Experimental results are given in Section 5, and, finally, some conclusions are drawn in Section 6.

2 Modeling Forms

In this paper, *Attributed Relational Graphs* (ARG, [13], [14]) are adopted for representing forms. The nodes describe objects or parts of objects, like lines, logos, instruction fields, and information fields, whereas the arcs describe the mutual geometrical relationships between the objects represented by the nodes. Let us state these concepts more formally.

DEFINITION 1. *Form Graphs.*

A form graph $\mathcal{FG} \doteq \{N, A, \mathcal{A}_N, \mathcal{A}_A, G_N, G_A\}$ *is an Attributed Relational Graph with nodes* N, *arcs* A, *node attributes in* $\mathcal{A}_N$, *arc attributes in* $\mathcal{A}_A$, *generating node attribute function* G_N, *and generating arc attribute function* G_A.

DEFINITION 2. *Node Attributes.*

Node attributes $\mathcal{A}_N \doteq \left\{\mathcal{A}_{N_i}, N_i \in N\right\}$ *are four-tuples*

$$\mathcal{A}_{N_i} \doteq \left\{R_i, T_i, D_i, S_i\right\} \tag{1}$$

where R_i *is a* registration flag, *computed by* $G_N^R(\cdot)$ *that indicates whether node* N_i *is used to register the form,* T_i *is the* type of the object *associated with node* N_i, D_i *is the* node description, *and* S_i *are the* search attributes *which are used for identifying the rectangular region* Γ_I *where the search must operate, and for specifying how the search*

2. In particular, INFORMys is an Open Look application developed on Sun Sparc machines.

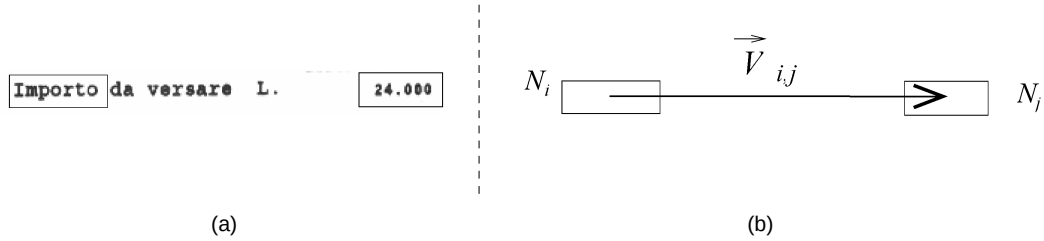


Fig. 3. Geometrical interpretation of an arc as a vector. (a) An instruction and a related information field. (b) The vector $\vec{V}_{i,j}$ connecting nodes N_i and N_j, corresponding to the two fields in (a).

takes place. Given a node N_i, function G_N generates the corresponding node attributes $\mathcal{A}_{N_i}$, that is the four-tuple $\{R_i, T_i, D_i, S_i\}$.

DEFINITION 3. *Node Type.*

Given $N_i \in N$, its type ranges in $T_i \doteq \{L, G, K, I\}$, that is, the type can be a line, a logo, an instruction field, and an information field, respectively, and the corresponding attribute is defined by the generating function $G_N^T(\cdot)$.

Lines act as reference points only, whereas all other types support information. Depending on the type of node, different descriptions arise which result in different attributes.

DEFINITION 4. *Node Description.*

Let $N_i \in N$ be a node. Depending on its type T_i, the node description is provided by function $G_N^D(\cdot)$ which generates the following attributes:

- *Line nodes.*
  $D_i \doteq \{Or_i, Th_i, Ln_i, St_i\}$. Or_i can only be H or V, which is for horizontal or vertical line, respectively. Th_i and Ln_i are the thickness and the length of the line (in pixels). Finally $St_i \in \{C, D\}$ is the line style, which indicates whether the line is continuous or dashed.
- *Logo nodes.*
  $D_i \doteq \{(Sx_i, Sy_i), Net_i\}$ (Sx_i, Sy_i) are the dimensions (in pixels) of the rectangle surrounding the logo. Net_i is a pointer to a set of parameters required by the logo recognition algorithm.
- *Instruction field nodes.*
  $D_i \doteq \{(Sx_i, Sy_i), Txt_i, Net_i\}$. (Sx_i, Sy_i) are the dimensions (in pixels) of the upright rectangle surrounding the word.[3] *Txt_i is the word textual information, that is, a string corresponding to the word ASCII coding. Net_i is a pointer to a set of parameters needed by the word recognition algorithm.*
- *Information field nodes.*
  $D_i \doteq \{Type_i, Min_i, Max_i\}$. $Type_i \in \{X, A, 9\}$ indicates whether the field is alphanumeric, alphabetic, or numeric. Min_i and Max_i indicate the minimum and maximum number of characters that can be found in the field.

DEFINITION 5. *Search Attributes.*

$S_i \doteq \{P_i, L_i\}$ are the attributes generated by $G_N^S(\cdot)$, which indicate where the search takes place (P_i) and provide information for its execution (L_i).

- *P_i: where the search takes place.*
  The searching area is a rectangle Γ_i, which can be identified in two different ways; the number of parameters that specify Γ_i indicates which description is used.
  - *absolute location of the searching area.*
    The searching area Γ_i is identified by $P_i \doteq \{x_A, y_A, x_B, y_B\}$, which represent the coordinates of its top-left (A) and bottom-right (B) corners.
  - *relative location of the searching area.*
    The searching area Γ_i is given by means of $P_i \doteq \{E_x, E_y\}$, which are the sides of Γ_i whose absolute position is given by one or two arcs.
- *L_i: how the search takes place.*
  *The $L_i \in \{U, D, L, R, C\}$ attribute describes how to search for the item into Γ_i. The search can in fact begin from the rectangle sides (U (*Up*), D (*Down*), L (*Left*), R (*Right*)) or from the center (C).*

Concerning the size of Γ_i, one can effectively deal with either fixed position items (e.g., information field N_9 in Fig. 4) or items that can be located in a variable position in a region of the form (e.g., instruction field N_7 in Fig. 4). In the first case, the Γ_i size needs to be only slightly greater than the corresponding item, whereas in the second case the search rectangle has to be large enough to cover the variable position of the item.

DEFINITION 6. *Arc Attributes.*

Let $A_{i,j}$ be the arc connecting node N_i to node N_j. The arc attributes $\mathcal{A}_A \doteq \left\{\mathcal{A}_{A_{i,j}}, A_{i,j} \in A\right\}$ are defined as the triple

$$\mathcal{A}_{A_{i,j}} \doteq \left\{R_{i,j}, \vec{V}_{i,j}, P_{i,j}\right\} \tag{2}$$

$\vec{V}_{i,j}$ is a vector, $P_{i,j}$ is the point where the vector is applied, and $R_{i,j}$ is the registration flag.

If $R_{i,j}$ = **`false`** (the arc $A_{i,j}$ is not used for registration), then only $\vec{V}_{i,j}$ is generated which represents a vector giving the relative position of Γ_j's barycenter once given Γ_i's (Fig. 3). The magnitude and orientation of $\vec{V}_{i,j}$ are denoted by $M_{i,j}$

3. Note that the word location algorithm recognizes also words of different size, but with the same aspect-ratio as that described in D_i.

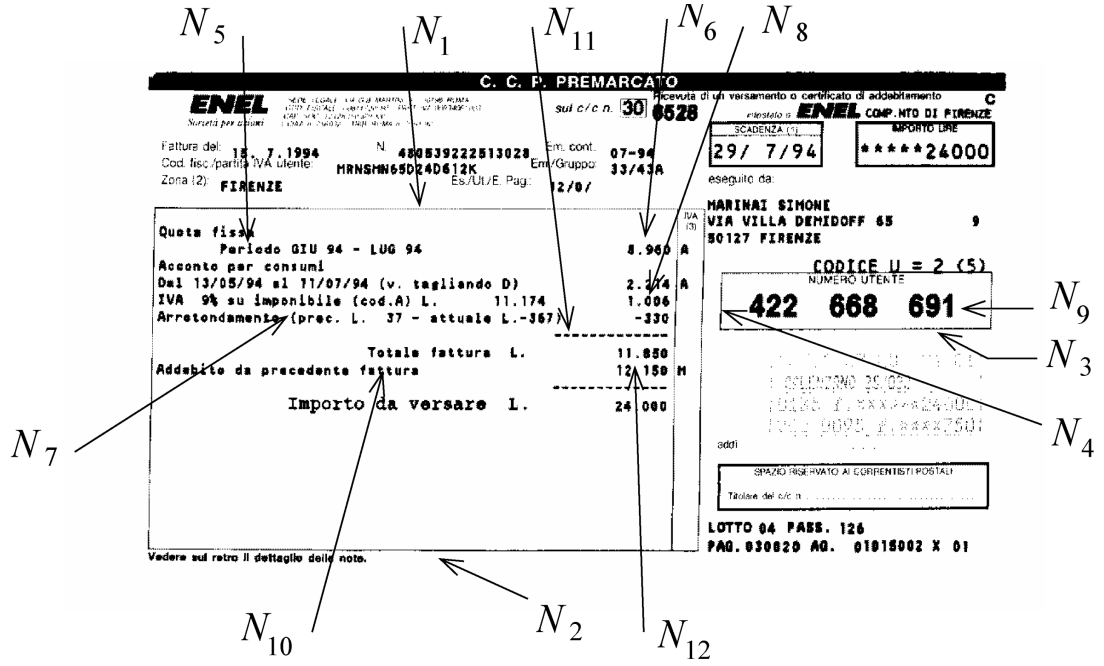


Fig. 4. An example of the description of a document. The pointed items (instruction fields, information fields, and lines) are used for creating the form graph described in Fig. 5.

and $\alpha_{i,j}$, respectively. In order to identify the case in which the location of an instruction field is carried out by means of two arcs, the vector length $M_{i,j}$ is left undefined since, in that case, only the direction of $\vec{V}_{i,j}$ is needed. The arcs can describe the mutual position of two nodes used to register the form or can be used to locate information fields. In the first case, $R_{i,j}$ = **true**, while in the second case, $R_{i,j}$ = **false**. As will be described in Section 3, our approach to form registration allows the user to describe the invariant part of the form layout by defining some registration landmarks that can be both lines and instruction fields. In other cases, the registration is constrained by considering the mutual position between registration landmarks. Nodes and arcs used to describe the registration landmarks and mutual positions give rise to a subgraph of $\mathcal{FG}$, that is referred to as the registration graph and is denoted by $\mathcal{FG}_r$. The registration graph plays a crucial role for the hypothesize and verification algorithm proposed in the following for registration.

DEFINITION 7. *Registration Graph.*

> *Given a form graph $\mathcal{FG}$, its registration graph $\mathcal{FG}_r$ only contains nodes and arcs used for the form registration, that is nodes and arcs for which the registration flag is* **true**. *$\mathcal{FG}_r$ is partitioned into two subgraphs $\mathcal{FG}_r^{(h)}$, and $\mathcal{FG}_r^{(v)}$, that will be used in the registration for the hypothesize and verification steps, respectively.*

Note that some nodes of $\mathcal{FG}_r$ can be used for form registration and also to locate information fields, whereas the arcs in $\mathcal{FG}_r$ are used only to register the form position.

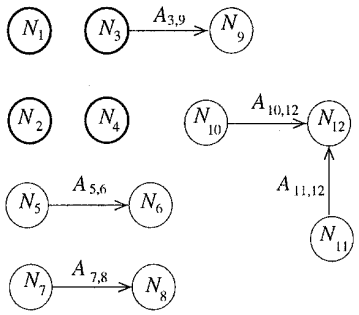
DEFINITION 8. *First Matching Item.*

> *The first matching item is the pair $I_{fm} \doteq \left\{\vec{V}_{fm}, P_{fm}\right\}$ used for firing the document registration.*

M_{fm} and α_{fm} denote the magnitude and the orientation of $\vec{V}_{fm}$, respectively. The registration transformation can be evaluated in three ways, as described by parameter $Type_{fm} \in \{1, 2, 3\}$. If $Type_{fm} = 1$, then I_{fm} is defined by a line node, N_1 $\left(\mathcal{FG}_r^{(h)} = \{N_1\}\right)$. P_{fm} is the center of the line, α_{fm} is aligned with the actual line, with direction corresponding to increasing coordinates, while $M_{fm} = 1$. If $Type_{fm} = 2$, then I_{fm} corresponds to an arc, $A_{1,2}$, and $\mathcal{FG}_r^{(h)} = \{N_1, N_2, A_{1,2}\}$. In the last case ($Type_{fm} = 3$), the point P_{fm} is obtained by considering the intersection of lines N_1, and $N_2\left(\mathcal{FG}_r^{(h)} = \{N_1, N_2\}\right)$, α_{fm} is aligned with the actual line corresponding to N_1, with direction corresponding to increasing coordinates, while $M_{fm} = 1$.

The form graph is constructed by the user who interacts with the INFORMys graphical user interface. In a typical session of graph building the user first loads a prototype form. Afterwards he selects the objects corresponding to the nodes, enters some attributes of the nodes, and describes the mutual relationships between objects. Here is a brief description of the steps required in order to insert graph items.

- Line node.
  The searching area for the line (P_i) is defined by drawing an upright rectangle on the prototype form. The user sets the registration flag and parameters Or_i, St_i, and L_i. An appropriate procedure is executed in



Nodes

$N_1 \equiv (\mathtt{t}, L, H, 1, 405, C, 40, 100, 458, 114, C)$
$N_2 \equiv (\mathtt{t}, L, H, 1, 405, C, 40, 354, 458, 368, C)$
$N_3 \equiv (\mathtt{t}, L, H, 1, 200, C, 459, 155, 667, 165, C)$
$N_4 \equiv (\mathtt{t}, L, V, 1, 41, C, 459, 156, 467, 205, C)$
$N_5 \equiv (\mathtt{f}, K, 20, 4, Periodo, Net_5, 47, 110, 210, 350, C)$
$N_6 \equiv (\mathtt{f}, I, 9, 3, 10, 80, 15, C)$
$N_7 \equiv (\mathtt{f}, K, 25, 4, imponibile, Net_7, 47, 110, 210, 350, C)$
$N_8 \equiv (\mathtt{f}, I, 9, 0, 10, 80, 15, C)$
$N_9 \equiv (\mathtt{f}, I, 9, 9, 9, 160, 20, C)$
$N_{10} \equiv (\mathtt{f}, K, 20, 4, Totale, Net_{10}, 47, 140, 250, 350, C)$
$N_{11} \equiv (\mathtt{f}, L, H, 1, 86, D, 365, 114, 450, 340, C)$
$N_{12} \equiv (\mathtt{f}, I, 9, 0, 10, 80, 15, C)$

Arcs

$A_{5,6} \equiv (\mathtt{f}, 295, 0)$
$A_{7,8} \equiv (\mathtt{f}, 288, 0)$
$A_{3,9} \equiv (\mathtt{f}, 21, \pi/2)$
$A_{10,12} \equiv (\mathtt{f}, 0, 0)$
$A_{11,12} \equiv (\mathtt{f}, 0, -\pi/2)$

Fig. 5. A typical $\mathcal{FG}$ associated with the items selected by the user during the form modeling phase of form shown in Fig. 4. The registration graph $\mathcal{FG}_r$, is composed of nodes N_1, N_2, N_3, N_4. A detailed description of the meaning of some nodes and arcs is given in the text.

order to locate all the lines in P_i, with orientation Or_i and style St_i. The detected lines are drawn by INFORMys on the prototype form, and the user interacts by identifying the correct one. Finally the system computes the appropriate values for Th_i and Ln_i from the selected line.

- Instruction field node and logo node.
  The values R_i, P_i, and L_i are entered in the same way as line nodes. Afterwards, the user draws a rectangular region surrounding the object to be located. The system evaluates Sx_i and Sy_i so as to identify the object to be used subsequently for learning the parameters of Net_i.
- Information field node.
  The values P_i and L_i are introduced as previously described. Attributes $Type_i$, Min_i, and Max_i are directly typed by the user.
- Arc.
  The definition of an arc is very simple. INFORMys shows all the nodes of the graph and the user is asked to select the two nodes which identify the arc, and to set the registration flag. All the other attributes are computed by the system on the basis of the positions of the objects corresponding to the connected nodes.

Fig. 5 represents the $\mathcal{FG}$ for the document depicted in Fig. 4, where the items corresponding to the nodes are indicated by arrows. Nodes N_1, N_2, N_3, and N_4 are used for registration and

$$Type^{fm} = 1 \left(\mathcal{FG}_r^{(h)} = \{N_1\}, \mathcal{FG}_r^{(v)} = \{N_2, N_3, N_4\}\right).$$

Examples of the meaning of some nodes and arcs are given in the following. N_1 describes a horizontal line with thickness one and length of 405 (pixels); the line must be found in region Γ_1, defined by points A = (40, 100) and B = (458, 114). N_8 describes an information field containing up to 10 digits and located into a rectangular region Γ_8 with E_x = 80 and E_y = 15. The location of the field is obtained by giving its position with respect to node N_7 by means of arc $A_{7,8}$. In order to take into account the variable position of the instruction field corresponding to node N_7, region Γ_7 is quite large (A = (47, 110), B = (210, 350)). Nodes N_{10} and N_{11} are used to locate the information field corresponding to node N_{12}. In the form of Fig. 4, neither N_{10} nor N_{11} contain enough information to locate N_{12}. Basically, N_{10} and N_{11} provide the horizontal and vertical position of the information field, respectively.

3 Form Registration

A typical problem that arises when acquiring documents with scanners is that there is a mismatch between the reference model and the incoming form, which is due to the difficulty of predicting exactly the location of the form on the scan bed. In the literature [2], the solution of this problem is referred to as form registration (or alignment), while the term skew denotes the rotation to which the document is subjected. Basically, we need to evaluate the roto-translation map which makes the incoming form congruent with the INFORMys model.

Many methods based on the location of single characters, have been proposed in order to detect the skew of unknown documents [20], [21]. Unfortunately, these methods are very time consuming since they do not take the knowledge of the form layout into account, and are not very suitable for registration. In order to register forms, a pursuable approach is that of locating lines and then use a sort of bottom-up scheme that requires processing the entire document. The solution proposed in [5] is limited to the estimation of the skew and is based on the detection of all the lines in the document, and then on a majority vote on line orientation. Although robust, that approach requires the analysis of the whole document and turns out to be useful when the form layout is not exactly defined, but we know that the form contains horizontal and/or vertical lines. In [3], the form registration is evaluated considering the relationships between the line corner junctions stored in the model and those found in the incoming form. This method is very accurate once working on tabular forms where there are many line crossings. When the form layout is exactly known, a bottom-up approach seems to be unnecessary. In some applications the form registration can be obtained simply by locating some specific items by hardcoding the form class knowledge [22]. Although very effective for customized applications, these methods have a very low flexibility. On the contrary, the approach proposed in [23], which is based on the ex-

traction of cross and end-point features, exhibits a higher flexibility. From the expected position of each feature in the document, the detector for that feature is applied to all pixels in an expanded region defined by the expected rotations and translations to which the form can be subjected. Geometrical constraints on the distance between features and the angles between pairs of features are used in order to make a registration hypothesis. Once a hypothesized transformation is found, it is verified considering a fixed number of additional landmarks. If verification is successful, then the document is rotated and translated accordingly. If the hypothesis fails, another one is generated and subsequently verified. This approach seems to be very effective since there is no need to extract all the features from the incoming document.

In INFORMys, we use a related approach in which lines, keywords, and logos are used to align forms. As shown in Section 2, the registration landmarks, which describe the invariant part of the layout, are stored as nodes of $\mathcal{FG}_r$. Moreover, instead of looking for any given item used for registration in the corresponding expanded region (as done in [23]), we define the expanded region only for the nodes of $\mathcal{FG}_r^{(h)}$ (see Definition 8), during the hypothesize phase. The other items are found by applying the hypothesized transformation to the location described in the attributes of $\mathcal{FG}_r^{(v)}$ nodes. The expanded region $\Gamma_i^E \left(N_i \in \mathcal{FG}_r^{(h)}\right)$ is specified by two error margins δ_x and δ_y, that give the maximum allowable displacement for the incoming form. Γ_i^E is chosen in such a way to take all the errors due to the positioning of the form into the scan bed into account.

The registration algorithm is based on the *hypothesize and verify* approach [15]. Basically, the main procedure (see Algorithm 1) is composed of two steps: hypothesis generation and verification. In hypothesis generation, a list $\mathcal{L}$ of transformations is found by aligning the first-matching item with a related item extracted from the incoming form $\mathcal{F}$. Each element of $\mathcal{L}$ is referred to as a hypothesis. When verifying a hypothesis $Hyp_j \doteq (\Theta_j, \Delta x_j, \Delta y_j)$, additional objects corresponding to the nodes $N_i \in \mathcal{FG}_r^{(v)}$ are searched in regions $\Gamma_{i,j}$ calculated according to transformation Hyp_j. This search is performed by function $\texttt{Verify} = \left(\mathcal{L}, \mathcal{FG}_r^{(v)}, \mathcal{F}\right)$, which, in turn, tests each hypothesis of $\mathcal{L}$ until one of them is successfully verified. If no hypothesis is successfully verified then the error margins (δ_x, δ_y) are increased by δ_{xi} and δ_{yi} so as to identify a larger region and hypothesis generation and verification is repeated until the maximum values $\left(\overline{\delta_x}, \overline{\delta_y}\right)$ are reached.

Algorithm 1 REGISTRATION-ALGORITHM

Input:

the registration graph $\mathcal{FG}_r \doteq \mathcal{FG}_r^{(h)} \cup \mathcal{FG}_r^{(v)}$;
the incoming form $\mathcal{F}$.

Output:

the roto-translation r which makes $\mathcal{FG}_r$ congruent with $\mathcal{F}$, $r = nil$ denotes that no registration is possible.

begin
$\delta_x \leftarrow \delta_{xo}$; $\delta_y \leftarrow \delta_{yo}$;
repeat
$\mathcal{L} \leftarrow$ **Hypothesize**$\left(\mathcal{FG}_r^{(h)}, \delta_x, \delta_y, \mathcal{F}\right)$
$r \leftarrow$ **Verify**$\left(\mathcal{L}, \mathcal{FG}_r^{(v)}, \mathcal{F}\right)$
$\delta_x \leftarrow \delta_x + \delta_{xi}$; $\delta_y \leftarrow \delta_y + \delta_{yi}$;
until $\delta_x > \overline{\delta_x}$ **or** $r <> nil$
end

In hypothesis generation and verification, when looking in $\mathcal{F}$ for objects that can correspond to node N_i some constraints must be satisfied. Two types of constraints can be defined: namely unary-constraints and binary-constraints [15]. With **unary-constraint**(N_i, O) we denote the function used to evaluate the constraint defined on N_i and O. The constraint is **true** (and we say that object O can match node N_i) if O is an object of type T_i and if the distances between numerical attributes of D_i and corresponding features of O are below a threshold defined in advance. Binary-constraints are applied to a pair of pairings of $\mathcal{FG}_r$ nodes and $\mathcal{F}$ objects. The nodes considered in the binary-constraint are connected by a $\mathcal{FG}_r$ arc. Let O and $\mathcal{U}$ be two objects of $\mathcal{F}$, **binary-constraint**$(N_i, N_j, O, \mathcal{U})$ is true if O can match N_i, $\mathcal{U}$ can match N_j, and the mutual position between the barycenters of O, and $\mathcal{U}$ (described by vector $\vec{V}_{O,\mathcal{U}}$) is compatible with $\vec{V}_{i,j}$. In hypothesis generation, $\vec{V}_{O,\mathcal{U}}$ is compatible with $\vec{V}_{i,j}$ if the distance between $M_{O,\mathcal{U}}$ and $M_{i,j}$ is below a threshold. The functions **unary-constraint** and **binary-constraint** are used in the following, in order to test whether an object can match a graph node.

The function **Hypothesize**, sketched in Algorithm 2, uses the following functions. **Expand**$(\Gamma_i, \delta_x, \delta_y)$ calculates the expanded region Γ_i^E. **FindObjs**$(\Gamma, T_i, D_i, S_i, \mathcal{F})$ locates (in region Γ) objects O for which **unary-constraint**(N_i, O) is true. For each object the coordinates of the barycenter, and its orientation (defined only for line nodes) are inserted into the output list. **LineAlign**, **ArcAlign**, and **CrossAlign** perform the alignment of I_{fm} with corresponding vectors extracted from $\mathcal{F}$. **LineAlign**$\left(\Gamma_1^E, D_1, S_1, \mathcal{F}\right)$ searches lines that can match N_1 in region Γ_1^E. For each line a transformation is calculated and inserted into the list $\mathcal{L}$. **ArcAlign**$(\mathcal{L}_1, \mathcal{L}_2, M_{1,2}, \alpha_{1,2})$ considers two lists ($\mathcal{L}_1$ and $\mathcal{L}_2$) of objects that can match N_1, and N_2. For each pair of objects $O \in \mathcal{L}_1$ and $\mathcal{U} \in \mathcal{L}_2$, such that **binary-constraint**$(N_i, N_j, O, \mathcal{U})$ is true, we obtain a hypothesis added to list $\mathcal{L}$. **CrossAlign**$(\mathcal{L}_1, \mathcal{L}_2, D_1, P_1, D_2, P_2)$ allows us to compute hypotheses by considering

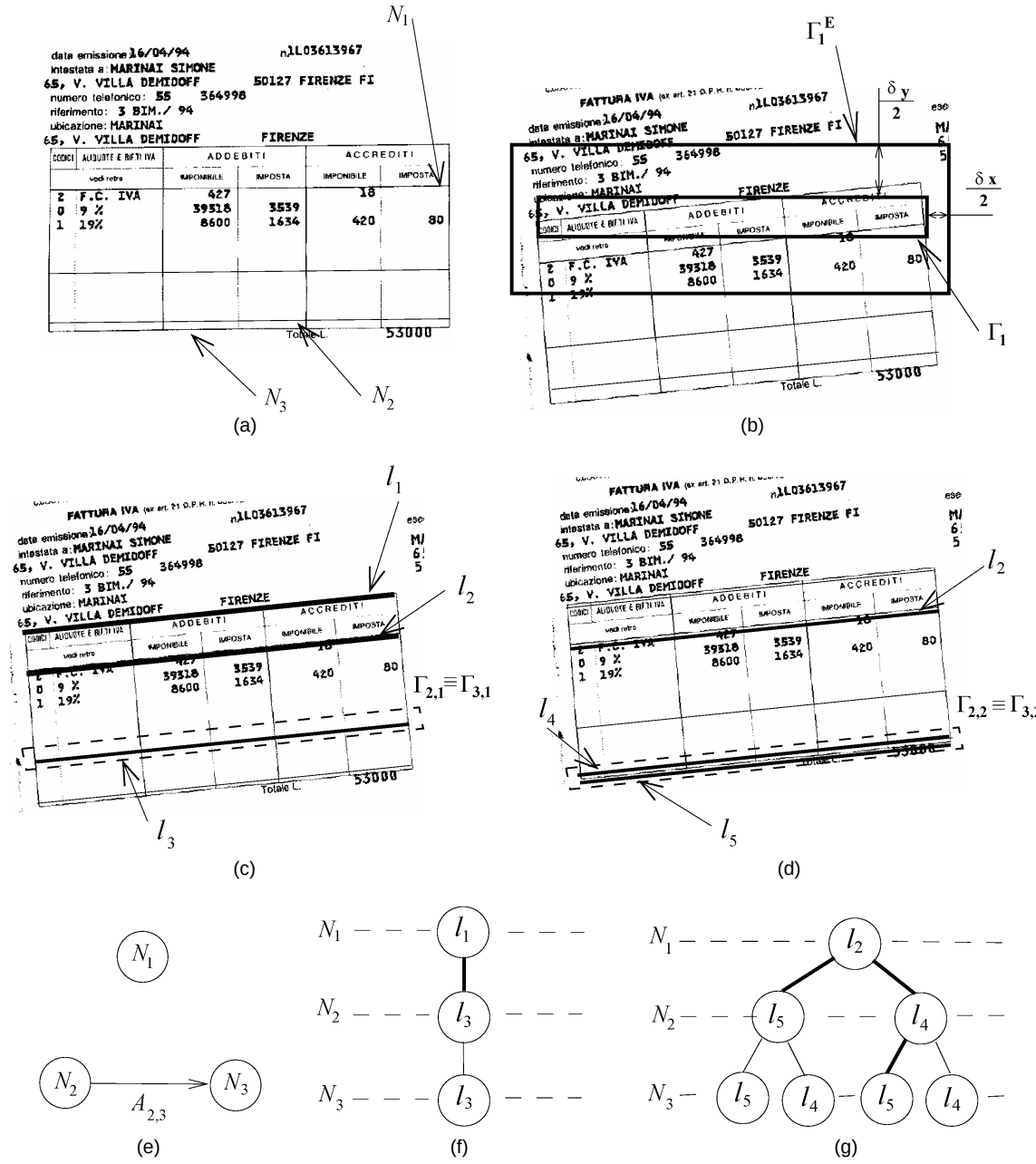


Fig. 6. Example for the registration algorithm. The registration graph, reported in (e), contains the horizontal lines pointed out in (a). The hypotheses are generated by using N_1 $\left(\mathcal{FG}_r^{(h)} = \{N_1\}\right)$, and $\left(\mathcal{FG}_r^{(v)} = \{N_2, N_3\}\right)$. In region Γ_1^E, depicted in (b), two lines can match N_1 (l_1 and l_2 in (c)). Hence two hypotheses (*Hyp*$_1$, and *Hyp*$_2$) must be verified. The verification of *Hyp*$_1$ is described in (c) and (f). In (c) we report the regions ($\Gamma_{2,1}$,$\Gamma_{3,1}$, that are superimposed) where to look for locating objects that can match $\mathcal{FG}_r^{(v)}$ nodes. Since the line l_3 is assigned to both N_2 and N_3, then the verification fails. In other words the verification tree, depicted in (f), does not contain a *consistent path*. The verification of *Hyp*$_2$ is shown in (d) and (g). In this case we can found two lines matching N_2 and N_3, and this corresponds to the *consistent path* l_2-l_4-l_5 in (g).

pairs of lines ($l_1 \in \mathcal{L}_1$, and $l_2 \in \mathcal{L}_2$). For each pair, the vector with the base at the intersection of l_1 with l_2 and with orientation defined by l_1 is aligned with I_{fm}, thus obtaining the hypothesis inserted into the list $\mathcal{L}$.

Algorithm 2 Procedure `Hypothesize`

Input:

$\mathcal{FG}_r^{(h)}$: the subgraph of $\mathcal{FG}_r$ used for hypothesis generation

the error margins δ_x and δ_y

the incoming form $\mathcal{F}$.

Output:

a list $\mathcal{L}$ of roto-translation transformations.

```
begin
  case Type_fm of
    1: begin
         Γ_1^E ← Expand(Γ_1, δ_x, δ_y);
         L ← LineAlign(Γ_1^E, D_1, S_1, F)
       end
    2,3: begin
         Γ_1^E ← Expand(Γ_1, δ_x, δ_y);
         L_1 ← FindObjs(Γ_1^E, T_1, D_1, S_1, F);
         Γ_2^E ← Expand(Γ_2, δ_x, δ_y);
         L_2 ← FindObjs(Γ_2^E, T_2, D_2, S_2, F);
         if Type_fm = 2
           then L ← ArcAlign(L_1, L_2, M_1,2, α_1,2)
         else L ← CrossAlign (L_1, L_2, D_1, P_1, D_2, P_2)
       end
  end
end
```

The **Verify** function tests the correctness of the hypotheses in $\mathcal{L}$, until a hypothesis is successfully verified or all the hypotheses are tested. A hypothesis Hyp_j is successfully verified if we can assign a distinct object to each of the $\mathcal{FG}_r^{(v)}$ nodes. We can assign an object O of $\mathcal{F}$ to node N_i, if O can match N_i, and for all arc $A_{h,i}\left(A_{h,i} \in \mathcal{FG}_r^{(v)} | h < i\right)$ **binary-constraint**(N_h, N_i, $\mathcal{U}$, O) = **true** (where $\mathcal{U}$ is the object assigned to node N_h). The behavior of the verification algorithm can be described graphically by means of a tree representation of the search space (the *verification tree*: $\mathcal{VT}$). The $\mathcal{VT}$ nodes contain one or more objects of $\mathcal{F}$. Each object is identified by the coordinates of its barycenter. For each hypothesis Hyp_j, a tree $\mathcal{VT}_j$ can be created. The hypothesis is represented in the root of $\mathcal{VT}_j$ that contains an object for each of the $\mathcal{FG}_r^{(h)}$ nodes. Each node at the ith level of $\mathcal{VT}_j$ contains an object of $\mathcal{F}$ that can match the ith node of $\mathcal{FG}_r^{(v)}$. The children of the nodes of the ith level correspond to the objects found by the algorithm when matching N_{i+1} of $\mathcal{FG}_r^{(v)}$ in region $\Gamma_{i+1,j}^E$. The $\mathcal{VT}_j$ leaves correspond to the last node of $\mathcal{FG}_r^{(v)}$. Given such a structure for $\mathcal{VT}_j$, hypothesis verification is based on the search of a *consistent path*. A *consistent path* is a path from the root to a leaf of the tree that satisfies the following two conditions:

- All the objects in the nodes of the path are distinct;
- Let O_i and O_j be the objects in the nodes of the path at levels i and j. Given an arc $A_{i,j}$ such that N_i and N_j are in the path, **binary-constraint**(N_i, N_j, O_i, O_j) = **true**.

The search of a consistent path is carried out by the depth-first traversal of $\mathcal{VT}$. The hypothesis is verified when a consistent path is found, otherwise the hypothesis is rejected. The actual behavior of the registration algorithm can be understood by the following example (Fig. 6).

EXAMPLE**.** In this example, the form registration is based on a registration graph (Fig. 6e) whose first matching item is a line node N_1. The registration graph $\mathcal{FG}_r$ is composed of line nodes N_1, N_2, and N_3. $Type_{fm}$ = 1, hence $\mathcal{FG}_r^{(h)} = \{N_1\}$, and $\mathcal{FG}_r^{(v)} = \{N_2, N_3\}$.

- **Hypothesis generation**. The first step is the expansion of region Γ_1 to Γ_1^E, where we look for I_{fm} (Fig. 6b). In order to evaluate the transformation required for form registration, a line that can match N_1 must be found in Γ_1^E. As we can see in Fig. 6c, two lines (l_1 and l_2) can potentially match line node N_1. The alignment of N_1 with l_1 and with l_2 yields two transformations denoted as Hyp_1 and Hyp_2, respectively.
- **Hypothesis verification**. The verification of hypothesis $Hyp_j (j = 1, 2)$ requires looking for a distinct line corresponding to N_i (i = 2, 3) into the region $\Gamma_{i,j}$ (see Figs. 6c and 6d). This can be also interpreted as looking for a *consistent path* in the verification tree (see Figs. 6f and 6g). In the case of Hyp_1 the line l_3 (Fig. 6c) can match N_2, but no line corresponding to N_3 can be found in region $\Gamma_{3,1}$. Hence the hypothesis verification fails, and we must verify the other hypothesis (Hyp_2). The verification tree for Hyp_2 is shown in Fig. 6g, where the *consistent path* (l_2–l_4–l_5) is reported. The path can be interpreted as the match of l_2 with N_1, l_4 with N_2, and l_5 with N_3, as shown in Fig. 6d. Note that the path (l_2–l_5–l_4) in Fig. 6g is not consistent since **binary-constraint**(N_2, N_3, l_5, l_4) is false.

This example is based on line nodes only, but logos or instruction fields can also be used for verification, thus giving rise to a similar behavior.

As already pointed out, the registration can also be based on an arc $A_{1,2}$ as I_{fm}, and the hypothesis is generated by using the function **ArcAlign**. In Fig. 7, an example is shown with the two items (N_1, N_2) used to determine arc $A_{1,2}$. Node N_1 corresponds to a logo, while N_2 corresponds to an instruction field. The hypothesized transformation is calculated by aligning the vector described in $\vec{V}_{1,2}$ with the vector connecting the barycenters of two objects that can match N_1 and N_2. As in the case of simple line nodes, regions Γ_1 and Γ_2 are expanded by the error margins (δ_x, δ_y).

4 LOCATION OF INFORMATION FIELDS

After form registration, the rototranslation transformation r can be used to locate the information fields in the incoming form $\mathcal{F}$. In INFORMys, there are essentially three ways of locating an information field corresponding to node N_k:

1) The information field is placed in a fixed position with respect to the layout.

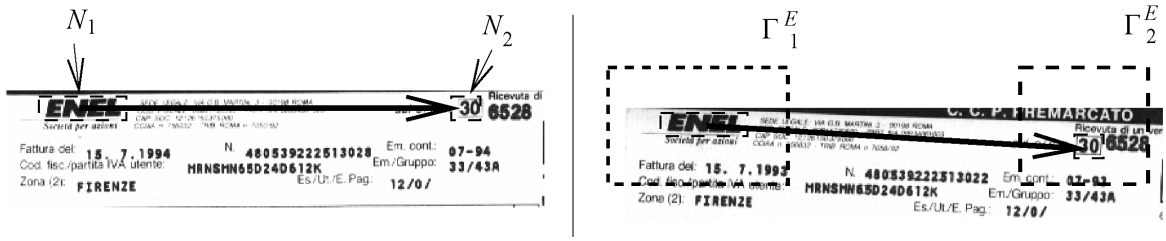


Fig. 7. A hypothesis generated by an arc connecting nodes N_1 and N_2 that are associated with a logo and an instruction field, respectively. In the hypothesis generation step, the expanded regions (Γ_1^E and Γ_2^E) are calculated for both the items linked by the arc.

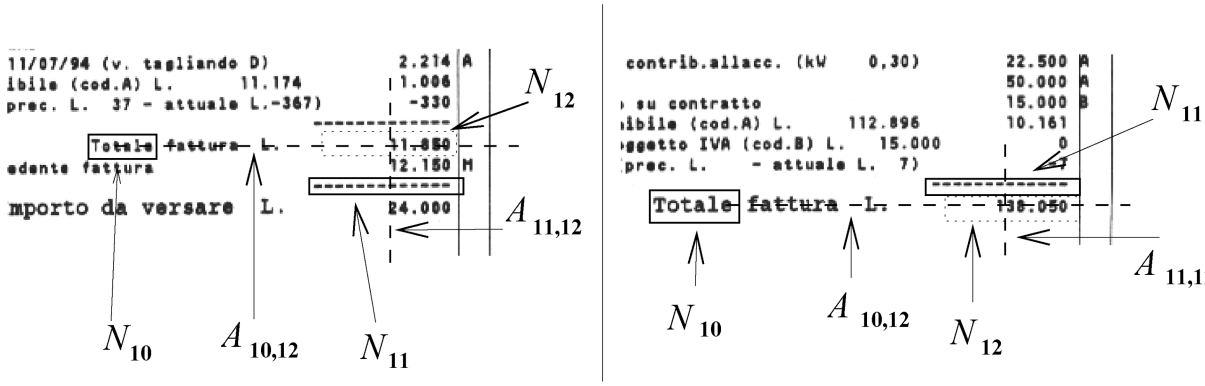


Fig. 8. Location of the information field N_{12} by giving its position with respect to two reference objects (line node N_{11} and instruction field node N_{10}). The nodes and arcs are part of $\mathcal{FG}$ depicted in Fig. 5. Note that the actual size of word *Totale* and horizontal distance from N_{10} to N_{12} change in the two forms.

2) The reference object and the corresponding information field are not in a fixed position in the form, but their mutual position is defined in advance.
3) The information field can be located by means of its relative position with respect to two reference objects, that are not in fixed positions in the form.

In the first case the location of the information field is obtained by applying the transformation r to points A (top left) and B (bottom right), described in search attributes of N_k ($P_k = \{x_A, y_A, x_B, y_B\}$). In the second and third cases, the information field can be found only after having located the corresponding reference objects, while the size of the search region is described by $P_k = \{E_x, E_y\}$. The number of reference objects depends on the number (na_k) of arcs having the "tip" in node N_k ($A_{i,j} \in \mathcal{FG} \,|\, j = k$). If $na_k = 1$, then the center of the rectangle containing N_k is found applying the vector $\vec{V}_{i,j}$ to the barycenter of the object corresponding to node N_i. If $na_k = 2$ $\left(A_{i_1,j_1}, A_{i_2,j_2} \middle| j_1 = k, j_2 = k\right)$, then the center of the rectangle containing N_k is defined by the intersection of the straight lines starting from objects matching N_{i_1} and N_{i_2} and with orientation α_{i_1,j_1} and α_{i_2,j_2}, respectively (Fig. 8).

In the second and third cases, the location of information fields requires, in general, to locate reference objects. Lines are located by an algorithm strictly related to the Run Length Smoothing Algorithm [24]), while the localization of logos and instruction fields is based on the algorithm described in the following.

4.1 Location of Logos and Instruction Fields

The location of logos and instruction fields is crucial for both form registration and location of information fields. In INFORMys, the recognition of the logos is based on a connectionist-based model, where a multilayer perceptron acting as an autoassociator is trained for each class. The neural networks are trained to reproduce the input layer to the the output layer. One hidden layer provides a compressed and nonlinear representation of the input information. It has been pointed out that reliable rejection criteria can be given for autoassociators that are based on the Euclidean distance between the inputs and the outputs [25], [26].

A massive experimentation of the multilayered autoassociators on the logo data base provided by the University of Maryland showed the effectiveness of the proposed approach also in presence of noise (see [27] for noisy logos created by Baird's model [28], and [29] for the case of spot-noise, in which documents contain blobs and strips that obstruct the logo partially).

Once properly segmented as a single pattern, each word corresponding to an instruction field (keyword) can be recognized by using the same connectionist architecture used for logos. In the first version of INFORMys, the keywords were in fact recognized using autoassociators only, but the subsequent massive experimentation on real-word data suggested us to exploit also the information that can be ex-

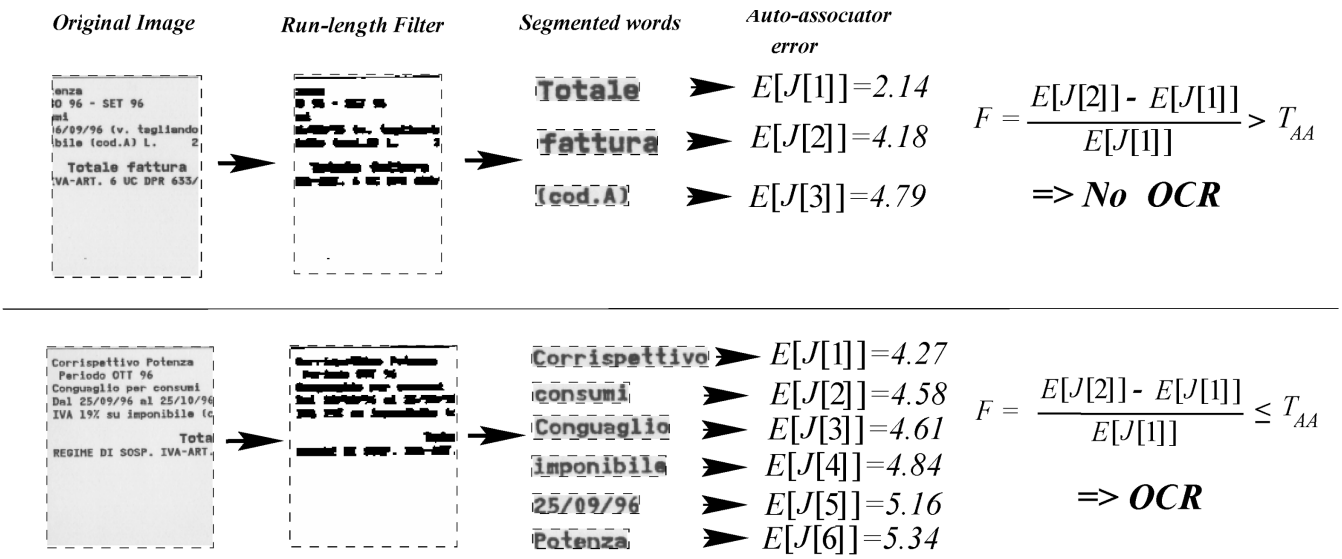


Fig. 9. Two examples of the location of the instruction fields *Totale* and *imponibile*. For each example, the figure contains the search region, the output of the run-length filter, the segmented words (with correct aspect-ratio), and the corresponding errors of the autoassociators. In the first example, $F > T_{AA}$. Hence, the prediction of the autoassociator is accepted. In the second example $F \leq T_{AA}$ and, therefore, the subsequent OCR-based check is required.

tracted from the recognition of the characters composing the keywords.

Algorithm 3 Function `RecWord`

Input:
The node N_i corresponding to a given instruction field
The incoming form $\mathcal{F}$.

Output:
the barycenter of the word corresponding to N_i. A *nil* value if the word was not found.

begin
$W \leftarrow$ `Rlsa`$(\mathcal{F}, \Gamma_i, Sx_i, Sy_i)$;
$E \leftarrow$ `AA`(W, Net_i);
$J \leftarrow$ `Order`(E);
if $\frac{E[J[2]]-E[J[1]]}{E[J[1]]} > T_{AA}$ **then** `RecWord` $\leftarrow W[J[1]]$
else begin
$j \leftarrow 0$;
repeat
$j \leftarrow j + 1$;
$d[j] \leftarrow$ `Warp`$(Txt_i,$ `OCR`$(W[J[j]]))$;
until j ≥ n **or** $d[j] \leq T_1$
if $j < n$ **then** `RecWord` $\leftarrow W[J[j]]$
else begin
$j_m \leftarrow \text{argmin}\{d[j], j = 1, \ldots, n\}$;
if $d[j_m] \leq T_2$ then `RecWord` $\leftarrow W[J[j_m]]$
else `RecWord` $\leftarrow$ *nil*
end
end
end

The location of the words corresponding to node N_i ($N_i \in \mathcal{FG} \mid T_i = K$) is described by function `RecWord` (see Algorithm 3), and Fig. 9 gives an example of how the algorithm operates. `RecWord` begins with the `Rlsa` (Run Length Smoothing Algorithm) which performs the extraction of regions in the incoming form $\mathcal{F}$ that are likely to contain a candidate word. For noisy forms and small character size, the gap between characters is sometimes filled by the noise and two consecutive characters are merged, while consecutive words are still separated. This means that it is very difficult to segment a word into characters, while it is relatively easy to segment a line into words by "smearing" the text. The classical run-length smoothing algorithm that we use looks for white spaces between black pixels on the same line and changes them to black if their length is less than a threshold (the threshold is chosen to be greater than the minimum intercharacter spacing). After merging characters, the positions of words are found by considering the sets of connected pixels with aspect-ratio nearly equal to Sx_i / Sy_i. The vector W contains up to n images extracted from the form by the run length smoothing algorithm.

These images are properly processed in order to feed the autoassociator Net_i, whose computation is carried out by function `AA` in Algorithm 3. The similarity of the word $W[j]$ according to N_i is based on the Euclidean distance between the input and output layers of the neural network Net_i. A vector E containing the Euclidean distance for each word represented in W is the output of function `AA`. The vector J, obtained by function `Order`, gives an ascending sorting of $E[j]$ ($E[J[1]] \leq E[J[2]] \leq \ldots \leq E[J[n]]$). In INFORMys the value $F = (E[J[2]] - E[J[1]]) / E[J[1]]$ is used to estimate the recognition confidence given by the autoassociator. If $F \leq T_{AA}$ then the autoassociator does not offer enough confidence for the decision. However, the error values give a significant measure of the similarity of the word $W[j]$ and the word modeled by N_i. The function OCR($W[j]$) performs the recognition of single characters on image $W[j]$ giving, as output, the ASCII code of the characters. `Warp` (X, Y) is a function that calculates the string edit distance of string X from string Y. The `Warp` function is based on a straightforward adaptation of the dynamic time warping algorithm [30] used for isolated word recognition, and takes into account insertions, deletions, and substitutions of characters. The distance $d[j]$ is calculated for each word until $d[j] \leq T_1$ or all

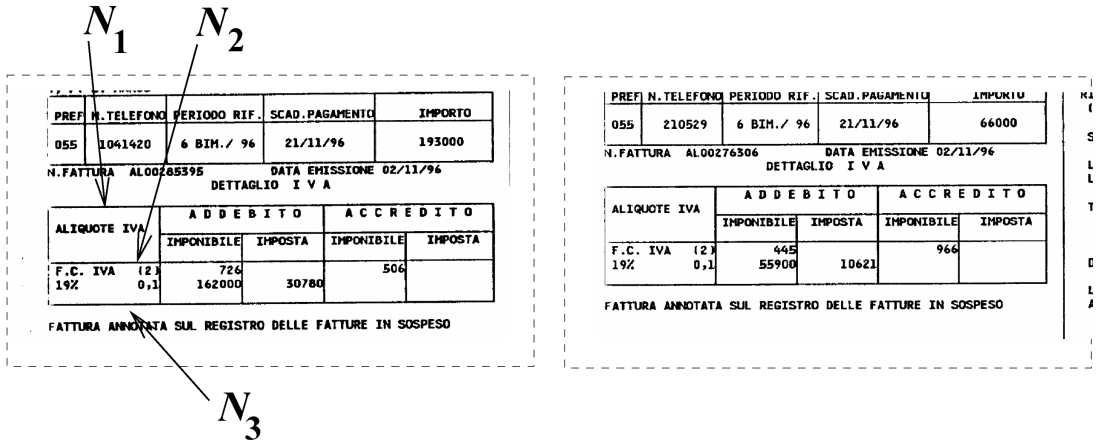


Fig. 10. Two examples of forms issued by Italian TELECOM. The registration graph $\mathcal{FG}_r$ has three nodes corresponding to the lines pointed in the the first form. Note that, due to the document displacement, the form in the right side of the figure does not contain the upper horizontal line.

the words are tested. The threshold T_1 is used to stop the OCR-based verification, whenever a word is close enough to Txt_i, which is the code of the string corresponding to reference word i. $W[J[j_m]]$ can be accepted as corresponding to the node N_i provided that $d[j_m] \leq T_2$, being T_2 the rejection threshold. Thresholds T_1 and T_2 are fixed on the basis of statistical analysis on the dictionary of words of the documents being processed. When no such knowledge is available, we can choose a severe criterion like $T_1 = 0$ and $T_2 = 0$, thus accepting only "exactly" recognized words, and rejecting all the others.

Algorithm 3 turns out to be a tradeoff between high accuracy and computational efficiency. A major assumption, that is confirmed by the experimental results in the next section, is that the algorithm is expected to rely mainly on the recognition of the autoassociator, that is in many cases there is no need to invoke the OCR module. It acts only in the case in which the prediction of the autoassociator is not reliable.

5 Experimental Results

In order to evaluate the performance of INFORMys, we carried out a massive experimentation of the system on real-world data. In this paper, we report the results of two experiments in which INFORMys is used for the automatic recognition of forms issued by Italian TELECOM and Italian Electrical company ENEL. These forms were kindly made available by the University of Florence.[4] The database for testing the system was composed of 100 TELECOM invoices and 182 ENEL invoices. In both experiments INFORMys was used for form registration, location, and recognition of information fields. The difference between these two applications is that in the first one the most critical point is the form registration, while in the second one the most critical point is the location of instruction fields.

INFORMys was running on a Sun ultra1 equipped with one cpu UltraSparc (143 MHz) and 64 KBytes of RAM.

5.1 TELECOM Forms: Test of the Registration Algorithm

In this experiment, the form registration was the most critical step since the invoices were manually placed in the scanbed without paying attention to their position.[5] As shown in Fig. 10, some forms were placed in the scanbed in such a way that the image is truncated. We tested the system using all the 100 black and white images acquired by a HP scanjet 4P scanner with a resolution of 300×300 dpi and size $1{,}201 \times 601$ pixels. The registration graph is composed of three horizontal lines N_1, N_2, and N_3 of the same length (see Fig. 10). Search regions for N_2 and N_3 were "small" rectangles (the size is 550×20 pixels for the horizontal line of 530 pixels) in order to speed up the search of lines and avoid confusion with other lines of the form. In all the experiments we used four different registration graphs $\mathcal{FG}_r^1, \ldots, \mathcal{FG}_r^4$ (see Table 1). These registration graphs differed only for the different searching areas Γ_i^E that were properly chosen in order to evaluate their effect on the system performance. The CPU time required by the registration algorithm depends in fact on Γ_1^E and on the number of increments of δ_x and δ_y that are required to locate N_1 (see Algorithm 1). It is worth mentioning that the CPU time should be mainly regarded as a way to show the different performance in the system when using various parameters. The best performances were found when adopting the form graph $\mathcal{FG}_r^1$, where Γ_1^E covers most part of actual lines corresponding to N_1. In this case Γ_1^E contains the line corresponding to N_1 for 88 of the 100 forms. Only for 10 forms δ_x and δ_y were increased once, while for the remaining two forms two region expansions were required. In $\mathcal{FG}_r^2$, a larger Γ_1^E is always used. This allows the system to avoid restarting the search, but the form registration requires more time (0.21 vs 0.55 second). The experiments using

4. These data are public domain and can be retrieved from the research group home page http://mcculloch.ing.unifi.it/~docproc.

5. Horizontal and vertical displacements with respect to the reference model were in ranges [–6, 6], [–17, 43] pixels, respectively.

TABLE 1
EXPERIMENTAL RESULTS FOR THE REGISTRATION ALGORITHM WHEN USING TELECOM FORMS

Form Graph	size of Γ_1^E (value of δ_{yo})	center of Γ_1	Mean CPU time (seconds)	N_{inc} 0	1	2	3
$\mathcal{FG}_r^1$	597 x 30 (δ_{yo} = 10)	280, 114	0.21	88	10	2	0
$\mathcal{FG}_r^2$	597 x 157 (δ_{yo} = 75)	280, 114	0.55	100	0	0	0
$\mathcal{FG}_r^3$	597 x 157 (δ_{yo} = 75)	280, 54	0.59	90	9	1	0
$\mathcal{FG}_r^4$	597 x 30 (δ_{yo} = 10)	280, 54	1.02	0	2	90	8

The four registration graphs are all equal apart from Γ_1^E, whose variable parameters are reported in the second and the third columns. The fourth column reports the mean CPU time required to register the forms. The last four columns report the number of forms which required the specified number of increments (N_{inc}) for δ_x and δ_y, (δ_{xi} = 15, δ_{yi} = 15). The size and center of Γ_i and the values of δ_{yo} are reported in pixel.

$\mathcal{FG}_r^3$ and $\mathcal{FG}_r^4$ showed that by choosing a completely wrong position and size for Γ_1^E, although more time consuming, the system can always register the form correctly. The location of the information fields, that are in fixed position, depends on the registration of the incoming form. Hence in all the tests the information fields where correctly found.

5.2 ENEL Forms: Location of Instruction Fields

In these experiments the documents were carefully placed on the scanbed, and, therefore, unlike the case reported in the previous subsection, the registration was not the major problem. For the recognition of the ENEL forms (Fig. 4) we used a form graph composed of eight nodes, which is a subgraph of that represented in Fig. 5, and a very simple registration graph with two nodes only (N_3, N_4). As described in Section 3, the intersection of the lines corresponding to these nodes and the line corresponding to N_3 are used for the registration ($Type_{fm}$ = 3). Three information fields were found by locating the three instruction fields described in nodes N_5, N_7, and N_{10}, corresponding to words *Periodo*, *imponibile*, and *Totale* respectively. Search regions for N_5 and N_{10} had a size of 120×170 pixels, while the search region for N_7 was 140×170 pixels.

The 182 images were acquired by using a scanner HP scanjet 4C with ADF (automatic document feeder). The resolution was 200×200 dpi with 256 gray levels, and the size of the images is $1{,}259 \times 708$ pixels. In these experiments, all the forms were correctly registered to the model and the first hypothesis generated by the registration algorithm was always the correct one and the major problem was that of locating the instruction fields.

In order to analyze the computational burden of the system for what concerns the location of instruction fields, it turns out to be useful to compare Algorithm 3 with the straightforward approach in which the words are simply recognized by OCR. Let N_i be the node corresponding to a generic instruction field located by means of function **RecWord**. Let $N_{i,j}^w$ be the number of words found by function **Rlsa** in region Γ_i of form $\mathcal{F}_j (j = 1, \ldots, m)$, $N_{i,j}^o$ be the number of calls to function **OCR** when locating word N_i in form $\mathcal{F}_j$, and define $N_i^w \doteq \sum_{j=1}^{m} N_{i,j}^w$, $N_i^o \doteq \sum_{j=1}^{m} N_{i,j}^o$. A rough estimation of the computational cost can be carried out by considering

$$R_1 \doteq \frac{N_i^o}{N_i^w},$$

which gives an idea of the number of calls to the OCR module in Algorithm 3. The parameter R_1, however, does not take into account the processing time of function **AA**. In our system, this function and the OCR recognition are carried out by artificial neural networks with multilayer architecture. Although the evaluation of function **AA** is based on autoassociators instead of the classical classifier-based mode used for OCR, in our experimental choices the computational complexity associated with the feedforward step of the two network is in fact roughly comparable. As a result the cost of the computation of the OCR module turns out to be **T(OCR)** $\simeq n \cdot$ **T(AA)**, where **T(AA)** is the computational cost for calculating $E[j]$ given word image $W[j]$. Hence, we can estimate the improvement obtained using our method by means of

$$R_2 \doteq \frac{N_i^w + N_i^o + Lm_i}{N_i^w \cdot Lm_i} \tag{3}$$

where Lm_i is the average length of the words extracted by function **Rlsa** in region Γ_i for all the m forms.

The parameter R_2 is reported in Table 2, for the three words used in the experiments and for different values of

TABLE 2
EXPERIMENTAL RESULTS ON THE ENEL FORMS CONCERNING THE LOCATION OF INSTRUCTION FIELDS

T_1	*Totale*	*Periodo*	*imponibile*	`total`
0.0	0.18	0.37	0.80	0.52
1.0	0.16	0.32	0.49	0.36
2.0	0.16	0.32	0.46	0.34
3.0	0.16	0.32	0.43	0.33

This table contains the values of R_2 corresponding to different values of T_1. The column `total` contains global values for R_2 corresponding to the location of the three words in all the 182 forms. For these experiments we choose T_{AA} = 0.3, T_2 = 3.0.

TABLE 3
PERFORMANCE ON ENEL FORMS
WHEN CHANGING THE THRESHOLD T_{AA}

		Totale	*Periodo*	*imponibile*	`total`
$T_{AA} = 0.1$	N_E	0	2	42	44
	N_R	0	0	1	1
	N_{AA}	182	139	23	344
	R_2	0.15	0.21	0.34	0.25
$T_{AA} = 0.2$	N_E	0	0	6	6
	N_R	0	1	2	3
	N_{AA}	180	112	3	295
	R_2	0.16	0.26	0.47	0.33
$T_{AA} = 0.3$	N_E	0	0	0	0
	N_R	0	1	2	3
	N_{AA}	177	61	0	238
	R_2	0.16	0.32	0.49	0.36
$T_{AA} = 0.4$	N_E	0	0	0	0
	N_R	0	1	2	3
	N_{AA}	171	41	0	212
	R_2	0.18	0.35	0.49	0.37
$T_{AA} = 0.5$	N_E	0	0	0	0
	N_R	0	1	2	3
	N_{AA}	156	16	0	172
	R_2	0.22	0.38	0.49	0.39

The number of errors is denoted by N_E, N_R is the number of rejected forms, and N_{AA} is the number of words located by using only the autoassociator. The experimental results corresponds to the choice of $T_1 = 1.0$ and $T_2 = 3.0$.

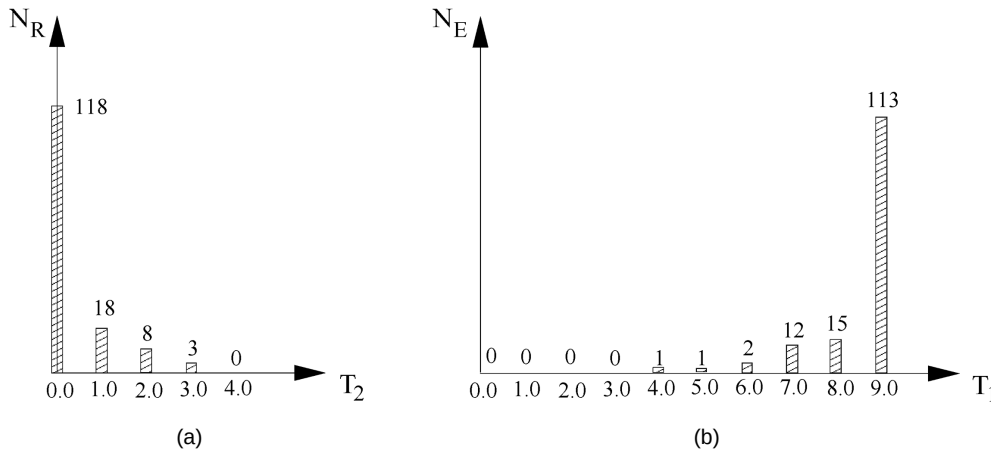


Fig. 11. Accuracy of the recognition of instruction fields in ENEL forms. The two plots report the number of rejected forms N_R and the number of errors N_E corresponding to different choices of T_2 and T_1.

threshold T_1. As we expected, the computational cost roughly approximated by R_2 decreases as T_1 increases. When choosing large values for T_1, Algorithm 3 often exits the check on the edit distance, whereas small values for the threshold represent a severe constraint which forces the algorithm to check all the candidate words. The best performance in terms of computational cost were achieved for word *Totale*, while the worst performance were found for the word *imponibile*.

Table 3 shows the performance on ENEL forms when changing the threshold T_{AA} which is used for deciding when the autoassociator-based criterion is itself sufficient to recognize the instruction fields. Table 3 shows that when choosing values for T_{AA} below 0.3 the accuracy is very poor. In that case, the scores of the first and second candidates are too close and the autoassociator-based criterion does not provide a reliable behavior. However, as can be seen in Table 3, a small increase of this threshold makes it possible to recover the errors, without a significant increase of the computational burden (see the parameter R_2). This is due to the important role played by the autoassociator neural network, that for values of $T_{AA} \simeq 0.3$ still limits significantly the calls to the OCR module. Note that the autoassociator neural network provides a significant contribution to the recognition process especially in the case of short words (e.g., compare the performance for the words *Totale* and *imponibile*).

As can be seen in Fig. 11, the recognition errors is strongly affected by the thresholds T_1 and T_2, but the be-

havior is quite regular and one can easily choose these parameters so as to have neither recognition errors nor rejection of forms (for instance, $T_1 = 1$ and $T_2 = 3.0$).

The number of rejected forms decreases when increasing the values of T_2 (plot A). We found that for values higher than 3.0 no words were rejected. On the other hand, the threshold T_1 plays a major role on the recognition error (see plot B). According to the previous analysis on the behavior of the autoassociator, $T_{AA} = 0.3$ was chosen for the experimental results reported in Fig. 11.

6 Conclusions

In this paper, we have described a flexible and efficient system for extracting information from images of financial documents like invoices and bills of service companies. The document layout is described by means of a model based on attributed relational graphs. This representation allows the user to specify the document structure by describing the features of the objects used for form registration and location of information fields.

We have proposed a form registration technique which is based on the hypothesize and verify paradigm. The proposed solution makes it possible to process a wide variety of layouts containing objects like lines, keywords, and logos. The location of keywords is based on a novel approach which integrates the recognition of whole words, based on autoassociator neural networks, with an OCR-based approach, where the words are recognized on the basis of the optimization of the string edit distance carried out by dynamic programming. The location of the information fields is based on the form model which makes it possible to determine these fields once the corresponding instruction fields are detected.

The overall performance of the system has been evaluated reporting two real-world applications for the recognition of invoices made available by the University of Florence. The experimental results reported in the paper support our claims concerning the flexibility (the two type of invoices are significantly different) and the efficiency of the overall system architecture (basically, the recognition process takes place in real-time on ordinary workstation).

Acknowledgments

We thank E. Francesconi, J. Sheng, M. Di Domenico, and F. Bocciarelli for the development of some important software modules of INFORMys. The recognition of instruction fields and logos was made possible by TRENNS Neural Network simulator. We are indebted to Marco Maggini (TRENNS designer) who offered his invaluable assistance for the usage of the simulator. This research was partially supported by MURST40%.

References

[1] Y.Y. Tang, C.Y. Suen, C.D. Yan, and M. Cheriet, "Financial Document Processing Based on Staff Line and Description Language," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 25, no. 5, pp. 738-753, 1995.

[2] R. Casey, D. Ferguson, K. Mohiuddin, and E. Walach, "Intelligent Forms Processing System," *Machine Vision and Applications*, vol. 5, no. 5, pp. 143-155, 1992.

[3] S.L. Taylor, R. Fritzson, and J.A. Pastor, "Extraction of Data From Preprinted Forms," *Machine Vision and Applications*, vol. 5, no. 5, pp. 211-222, 1992.

[4] T.M. Ha and H. Bunke, "Model-Based Analysis and Understanding of Check Forms," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 8, no. 5, pp. 1,053-1,081, 1994.

[5] S.W. Lam, Javanbakht, and S.N. Srihari, "Anatomy of a Form Reader," *Proc. Second Int'l Conf. Document Anal. Recog.*, pp. 506-509, Tsukuba, Japan, 1993.

[6] T. Watanabe, Q. Luo, and Sugie, "Layout Recognition of Multikinds of Table-Form Documents," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 432-445, 1995.

[7] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis," *Proc. IEEE*, vol. 80, pp. 1,079-1,092, 1992.

[8] S.W. Lam and S.N. Srihari, "Multi-Domain Document Layout Understanding," *Proc. First Int'l Conf. Document Anal. Recog.*, pp. 112-120, 1991.

[9] D.S. Doermann and A. Rosenfeld, "The Processing of Form Documents," *Proc. Second Int'l Conf. Document Anal. Recog*, pp. 497-501, Tsukuba, Japan, 1993.

[10] T. Watanabe, Q. Luo, and Sugie, "Structure Recognition Methods for Various Types of Documents," *Machine Vision and Applications*, vol. 6, no. 6, pp. 163-176, 1993.

[11] J. Yuan, L. Xu, and C.Y. Suen, "Form Items Extraction by Model Matching," *Proc. First Int'l Conf. Document Anal. Recog.*, pp. 210-218, St. Malo, France, 1991.

[12] J. Yuan, Y.Y. Tang, and C.Y. Suen, "Four Directional Adjacency Graphs (fdag) and Their Application in Locating Fields in Forms," *Proc. Third Int'l Conf. Document Anal. Recog.*, pp. 752-755, Montreal, Canada, 1995.

[13] M.A. Eshera and K.S. Fu., "A Graph Distance Measure for Image Analysis.," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 14, no. 3, pp. 398-408, 1984.

[14] M.A. Eshera and K.S. Fu., "An Image Understanding System Using Attributed Symbolic Representation and Inexact Graph-Matching.," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 5, pp. 604-617, 1986.

[15] W.E.L. Grimson, *Object Recognition by Computer, the Role of Geometric Constraints*. MIT Press, 1990.

[16] F. Cesarini, S. Marinai, and G. Soda, "Object Registration for Visual Inspection Operations," *Proc. Int'l Conf. IECON 94*, pp. 987-993, 1994.

[17] T.K. Ho, J. Hull, and S.N. Srihari, "A Computational Model for Recognition of Multifont Word Images.," *Machine Vision and Applications*, vol. 6, no. 6, pp. 157-168, 1993.

[18] S.X. Zhao and S.N. Srihari, "A Word Recognition Algorithm for Machine-Printed Word Images of Multiple Fonts and Varying Qualities," *Proc. Third Int'l Conf. Document Anal. Recog.*, pp. 351-354, Montreal, Canada, 1995.

[19] Y. le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, pp. 541-551, 1989.

[20] S. Hinds, J. Fisher, and D. D'Amato, "A Document Skew Detection Method Using Run-Length Encoding and the Hough Transform," *Proc. 10th Int'l Conf. Pattern Recognition*, pp. 464-468, 1990.

[21] L. O'Gorman, "The Document Spectrum for Page Layout Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1,162-1,173, 1993.

[22] M.D. Garris, J. Blue, G. Candela, D. Dimmick, J. Geist, P. Grother, S. Janet, and C. Wilson, "Nist Form-Based Handprint Recognition System. Nistir 5469," *U.S. Dept. of Commerce, Technology Administration, Nat'l Inst. Standards and Technology*, 1994.

[23] D.S. Doermann, "Document Image Understanding: Integrating Recovery and Interpretation," PhD thesis, Univ. Maryland, College Park, 1993.

[24] Y.Y. Tang, C. Yan, and C.Y. Suen, "Document Processing for Automatic Knowledge Acquisition," *IEEE Trans. Knowledge and Data Engineering*, vol. 6, no. 1, pp. 3-20, 1994.

[25] M. Bianchini, P. Frasconi, and M. Gori, "Learning in Multilayered Networks Used as Autoassociators," *IEEE Trans. Neural Networks*, vol. 6, no. 2, pp. 512-515, 1995.

[26] M. Gori, L. Lastrucci, and G. Soda, "Autoassociator-Based Models for Speech Verification," *Pattern Recognition Letters*, vol. 17, pp. 241-250, 1996.
[27] F. Cesarini, M. Gori, S. Marinai, and G. Soda, "A Hybrid System for Locating and Recognizing Low Level Graphic Items," *Graphics Recognition*, R. Kasturi and K. Tombre, eds., *Lecture Notes in Computer Science*, pp. 135-147. Springer Verlag, Mar. 1996.
[28] H.S. Baird, "Document Image Defect Models," *Structured Document Image Analysis*, pp. 547-555, Springer-Verlag, 1992.
[29] F. Cesarini, E. Francesconi, M. Gori, S. Marinai, J. Sheng, and G. Soda, "A Neural-Based Architecture for Spot-Noisy Logo Recognition," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 175-179, Ulm, Germany, Aug. 1997.
[30] H. Sakoe and C. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," *IEEE Trans. Acoustic Speech and Signal Processing*, vol. 26, pp. 43-49, Feb. 1978.

Francesca Cesarini received her degree in Mathematics from the University of Florence, Italy, in 1968. She became a researcher at the National Council of Research in 1971, and, since 1983, she has been an associate professor of computer science at the Department of Systems and Computer Science of the Florence University. Her research interests include data structures and algorithms for database management systems, object data models, and techniques for document representing and understanding.

Marco Gori received the laurea in electronic engineering from Università di Firenze, Italy, in 1984, and the PhD degree in 1990 from Università di Bologna, Italy. He was also a visiting student at the School of Computer Science, McGill University, Montreal. In 1992, he became an associate professor of computer science at Università di Firenze and, in November 1995, he joined the University of Siena. His main research interests are in pattern recognition, neural networks, and artificial intelligence. In these fields, he has contributed to the organization of several scientific events, including the recent International Summer School on "Adaptive Processing of Sequences and Data Structures," whose lectures have been published in a book coedited with C.L. Giles.

Dr. Gori serves as a program committee member of several workshops and conferences mainly in the area of neural networks and pattern recognition. He acted as guest coeditor of the Neurocomputing Journal for the special issue on recurrent neural networks (July 1997). He is an associate editor of the IEEE Transactions on Neural networks, Neurocomputing, and Neural Computing Survey. He is the Italian chairman of the IEEE Neural Network Council (R.I.G.) and is a member of the IAPR, SIREN, and AI*IA Societies. He is also a senior member of the IEEE.

Simone Marinai received the Laurea in electronic engineering in 1992, from Università di Firenze, Italy. He obtained the PhD degree in computer science in 1996 defending a thesis on the extraction of information from structured documents. He is currently a Postdoctoral fellow at Dipartimento di Sistemi e Informatica, Università di Firenze. His research interests include document analysis and understanding, artificial intelligence, neural networks.

Giovanni Soda received his degree in mathematics from the University of Florence, Italy, in 1969. In 1971, he was a researcher at the national Council of Research, where his activity included formal systems for language manipulation. Since 1975, he has been at the University of Florence, where he is presently an associate professor of artificial intelligence at the Department of Systems and Computer Science (DSI) of the University of Florence. His current interests include Knowledge Representation Systems, integration of artificial intelligence techniques with Neural Networks, Document Processing. He was the general chairman of the AI*IA95 held in Florence in 1995. He is a member of the steering committee of AI*IA, the Italian Association of Artificial Intelligence and is a member of the IEEE, ACM, and IAPR Societies.