DOI: 10.15439/2015F383
ACSIS, Vol. 5

Logical Structure Recognition of Diagram Images

Jerzy Sas
Wroclaw University of Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: jerzy.sas@pwr.edu.pl

Urszula Markowska-Kaczmar
Wroclaw of Technology,
Wyb.Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: urszula.markowska-kaczmar@pwr.edu.pl

***Abstract*—This document presents a method of a logical links structure recognition between elements on diagrams. The applied approach intuitively mimics a human way of recognition that relies on merging already found connectors into more complex ones. This procedure is modeled by our method where simple and obvious connectors and gradually extended to more complex structures. Each iteration may lead to modification of connectors set obtained so far. The modifications are managed by a rules set describing logical and graphical constraints that should be satisfied by the connectors structure. If the extension leads to violation of constraints defined by the rules then the modification is not carried out. In this way, the recognized diagram structure is consistent with the assumed principles. The method was experimentally validated using the set of diagrams from three domains. In conclusions, method's advantages and drawbacks are discussed.**

I. Introduction

IN the last years there is a big interest in similar content image retrieval. There is plenty of research in this area. A deep survey is presented in [1]. The similar content image retrieval can be very helpful in automatic image annotation, in story illustration, copy detection, web image search and art image analysis.

It can be also helpful in searching similar text documents if they contain images. Usually, images illustrate the content of document. They contain information included in document in condensed form and there is less problems with unambiguous comparison of expressed idea. This explains a strong interest in its application to similar patents search in order to speed up the procedure of patenting and to protect intellectual property rights. For automatic querying it is necessary to convert the information in the images into a high-level description. Usually, in the case of technical documents, images represent engineering drawings, diagrams, algorithms, operations and processes shown as charts. Images of this kind present the structure consisting of certain elements and connections between them. Automatic recognition of the connections structure is the first step towards further automatic analysis or even machine understanding of images. The problem is important and challenging, because the documents have highly-complex structure, tabular and graphical information is embedded and they contain conflicting technical jargon. Processing embedded images can aid to solve the problem.

Typically, to apply such approach it is necessary to find images in the whole document. Then, images are processed in order to classify them to various classes: charts, diagrams, schemes, flowcharts, plots and photos. Next, a method dedicated to a given class of images is applied in order to recognize particular elements and their interconnections.

It is worth noting that such image interpretation allows to write the content of an image in the electronic form, which facilitates its storage and further processing and comparison.

In our research we focus on connectivity of elements in diagrams and flowcharts. The methods dedicated to this kind of graphics have to find: a) types of elements shown (various depicted shapes), b) segments of lines not belonging to found shapes, and c) connections created by these segments. Because diagrams usually contain texts embedded in diagram elements, it is also necessary to detect text areas, recognize it by applying OCR techniques and finally assign recognized texts to graphical diagram elements.

The aim of the research described in this paper was to find a method that is able to retrieve logical links between elements depicted in the diagram. This logical structure is then expressed in an XML file. It is a difficult task especially when we consider that one connection may exist between more than two elements and the line segments constituting connectors can intersect.

The paper consists of six sections. The next section describes related works. Section III formulates problem to solve. The subsequent section presents the developed method. Section V experimentally validates our approach. Finally, some conclusions and recommendations related to further works are presented.

II. Related Works

Early survey of works in this area is described in [2]. The author writes that diagram recognition faces many challenges, including the great diversity in diagrammatic notations, and the presence of noise and ambiguity during the recognition process. Despite the flow of time from the year of this publication, all mentioned above features characterizing diagram interpretation constantly cause problems in chart recognition now.

The paper [3] reviews research from the last decade. The authors present the whole process of chart recognition: chart segmentation, chart classification, chart interpretation and discuss existing solutions.

Relatively many works are devoted to online flowcharts recognition. In [4], the analysis to label each stroke of the flowchart and to group the strokes depending on the symbol they belong to is presented. The same area of research is represented in the paper [5]. In this paper the search for a suitable interpretation of the input is formulated as a combinatorial optimization task containing the max-sum problem. The recognition pipeline consists of two main stages. First, groups of strokes possibly representing symbols of a sketch (symbol candidates) are segmented and relations between them are detected. Second, a combination of symbol candidates best fitting the input is chosen by solving the optimization problem. The work [6] also concerns online charts but is focused on hand-drawn electric circuit diagram recognition using 2D dynamic programming. The paper [7] presents another approach to hand drawn organizational diagrams that is based on Bayesian conditional random fields (BCRFs) that jointly analyzes all drawing elements in order to incorporate contextual cues. The classification of each object affects the classification of its neighbors. BCRFs allow flexible and correlated features. The online recognition of diagram is mainly applied in order to automatically check student tests.

Currently there is a great interest in flowchart recognition in the context of patent search. The paper [8] describes measures for assessing the effectiveness of flowchart recognition methods in the context of patent-related use cases. A survey of approaches can be found in [9]. A system for semi-automatic chart ground truth generation is introduced in the paper [10]. Using the system, the user is able to extract multiple levels of ground truth data.

Some works are devoted to chart recognition in documents. They apply various classification methods. In [11] spiking neural networks are used. The paper [12] presents a system for recognizing a large class of engineering drawings characterized by alternating instances of symbols and connection lines. The class of considered images includes domains such as: flowcharts, logic and electrical circuits, and chemical plant diagrams. The output of the system includes a list identifying the symbol types and interconnections. It may be used for design simulation or as a compact portable representation of the drawing. The method consists of two steps. First, domain independent rules are used to segment symbols from connection lines in the drawing image that has been thinned, vectorized, and preprocessed in routine ways. Then a drawing understanding subsystem works together with a set of domain-specific matchers to classify symbols and correct errors automatically. They also proposed an interface to correct residual errors interactively.

Another important problem in diagram recognition and its automatic interpretation is recognition of texts appearing on diagrams. Separation of textual and graphical layers simplifies the further diagram structure analysis by reducing a number of involved graphical elements. It also makes it possible to attach textual information attributes to detected graphical elements of the diagram. In our approach we used the text separation method described in [13]. The method consists of three stages. In the first stage the text region candidates are elicited based on connected components analysis and some simple geometrical properties of connected components clusters. At the second stage, pattern recognition methods are applied to the set of candidates to discriminate between true text areas and other "false" candidates. Finally, OCR is applied to candidate regions and the final text region set is refined based on the analysis of the contents of the OCR-recognized strings.

III. Problem formulation

In the further part of this article by a *diagram* we will mean a drawing that shows a set of entities - *diagram elements* (DEs) and connections between them. In the literature, the term "diagram" is used interchangeably with "chart", but diagram seems to be more general. We will be considering diagram images created with appropriate software or precisely drawn manually using drawing tools (rulers, drafting templates) and then converted into raster images by scanning. Hand-sketched diagrams are out of the scope of this article due to big inaccuracies appearing in this type of drawings. Our aim is to retrieve the logical structure of the diagram, so that it corresponds to intents of the diagram author. Informally, by the logical structure of the diagram we mean here the links between diagram elements. Diagram elements represent various items appearing in the real world modeled by the diagram. Their meaning depends on the domain of application. In the case of program flowcharts they can represent: statements, code blocks, conditions, data sources etc. In a logical circuit diagram they represent gates and functional blocks like: registers, multiplexers, flip-flops, etc. In organizational charts their elements are usually officials or departments. Although the method described here can be applied to any kind of diagrams, we mainly focus on organizational charts and program flowcharts. The element of the diagram is depicted by a simple 2D geometric shape like: rectangle, circle ellipse, rhombus, diamond. There are some attributes assigned to diagram elements. The basic attributes of an element are the kind of 2D shape and the textual description (the text usually inscribed into the element shape). Additional attributes that can be easily retrieved from the diagram image are: the shape interior color, shape line color and the shape border line width. They can be meaningful in certain types of diagrams, in other types of diagrams they may be ignored. The methods used to recognize shapes appearing in diagrams and to evaluate their attributes will be shortly described in the section IV-I.

The link in the diagram represents a logical relation or an association between DEs. Links are graphically represented by polylines or sets of intersecting or connected polylines, which endpoints are in the close vicinity of DEs being connected. Depending on an application domain, various types of links can be distinguished. The simplest links are one-to-one links which represent the association between a pair of DEs. The many-to-many link is the more complex case that associates the larger set of DEs. The links can be undirected or directed. In directed links some polyline endpoints are arrows. The directed link usually indicates the information flow or organizational subordination. If directed links are used the many-to-many association may turn into one-to-many association where only the single endpoint of the link is not arrowed, while all remaining endpoints are arrows. The graphical representation of the link will be further called *connector*. The connector is therefore the set of polylines that intersect each other or constitute T-style junctions. Finding connectors that are known to represent one-to-one links is a simple technical problem. Also in the case where it is known that the connection of polylines belonging to a common connector is indicated by dots (or other graphical marks) placed at connection points, the problem only lies in reliable connection marks recognition. It is however much more complicated if we cannot assume that connections of polylines belonging to a connector are graphically marked by connection marks. In such case the recognition of diagram structure must be based on the trial to "guess" the diagram author intention. In the further part of the paper, we will describe the method that deals with this kind of diagrams.

We are considering here the method which starts with vectorized diagram image on its input. The vector representation of the original (raster) image is obtained by applying the sequence of image processing operations followed by the vectorization procedure that converts a binary image into the set of line segments (vectors). Let us also assume that DEs have been already successfully found. In our approach we used the shapes recognition method based on vector sequence matching to basic shapes defined either by rules or algebraically, as described shortly in section IV-I. The vectors constituting found DEs were identified and extracted from further considerations. Let $E = \{e_1, e_2, ..., e_N\}$ denote the set of found DEs and let L denote the set of line segments (called later *edges*) not assigned to any detected DE. The edge is a pair of 2D points being its ends on the plane $l_j = (p_{0j}, p_{1j})$, $p_i = (x_i, y_i)$. Line segments in L possibly belong to connectors. Formally, a connector is a subset of connected edges from L. Our aim at this stage is to gather as many as possible edges into disjoint subsets $C_i \subseteq L$ corresponding to connectors, while leaving as little as possible lines unassigned, i.e. belonging to the unassigned set U. The result is the family of subsets $\{C_1, C_2, ..., C_M\}$. The construction of connector sets can be considered as a rule-based process, where rules define some constraints that must be satisfied in order to put a certain subset of edges into a connector, as well as principles that force inclusion of some edges in a single connector. Rules determining principles of reasonable construction of connectors were derived from the analysis of diagram structures appearing on typical diagram images. The analysis has been carried out using the set of diagram images from various domains, that we used for testing of our diagram analysis methods. The rules defining the construction of a connector C_i of L are as follows:

- the elements of C_i are coherent (i.e. for each pair of edges in C_i there is a sequence of other edges in C_i, possibly empty, that connects them);
- if there is an edge in C_i that has a vertex not shared with other lines in C_i (i.e. it is the endpoint of a polyline which elements are within C_i) then it must be close to one of detected DEs from the set E, such a vertex is a *terminal vertex*;
- there are no two vertices of edges in C_i that are terminal vertices and are close to the same DE (the connector consisting just of a single edge cannot link the DE with itself);
- there are no cycles in the graph defined by the set C_i, i.e. there are no polylines in C_i that intersect with themselves;
- if there is an edge l_a in C_i that has a common vertex with another edge l_b in L and l_b is not assigned to any other connector C_j then l_b must also belong to C_i (no connectors ending in the middle of polylines);
- if p is an internal vertex shared by two edges belonging to C_i then it cannot be closer than ϵ to an internal vertex shared by two edges belonging to another connector C_j (connectors cannot touch each other, except of the case that the terminal edge endpoint touches a terminal edge endpoint of another connector);
- each edge is uniquely assigned to one of subsets C_i or to U (edges are not shared by connectors);
- the longest path connecting two terminal vertices in C_i cannot be longer than assumed threshold i.e. 8 (no connectors of very complicated shape);
- the angle between two adjacent edges in C_i which do not share the common vertex with any other edge within the same connector is not smaller than the right angle (no acute angles in the polyline segments of connectors);
- width of each edge within C_i does not differ by more than 30% from the weighted average line width in the connector;
- two elements $e_i, e_j \in E$ cannot be connected along more than one path within the single connector (they can be however connected by many paths, provided that they belong to different connectors).

Because it seems reasonable to merge connectors into more complex ones, as far as it does not lead to violation of rules presented above, then the ultimate aim is to partition the set L into the family of subsets $\{U, C_i : i = 1, ..., M)\}$ so as to minimize the number of subsets C_i with the additional constraint that the set U does not contain any subset of edges that constitutes a valid connector.

For practical purposes related to automatic analysis of the diagrams it is essential not just to find connectors (being the sets of edges) but rather to determine the sets of connected DEs which are connected by individual connectors. Therefore, the final result of the structure recognition procedure is the family of sets of DEs, where each set defines elements connected by the single connector. Additionally, because each connector endpoint can be marked with an arrow, this information should be also retrieved and included in the data structure being the output of the recognition procedure. Each endpoint of the connector is described by a pair (e, t) where $e \in E$ and $t \in \{true, false\}$ indicates whether or not the endpoint of the connector linked to e is an arrow. The elements connected by the single connector C can be specified by the multiset $V_C = \{(e_i, t_i) : e_i \in E, t_i \in \{true, false\}\}$. The expression $t = true$ denotes the arrow appearance and $t = false$

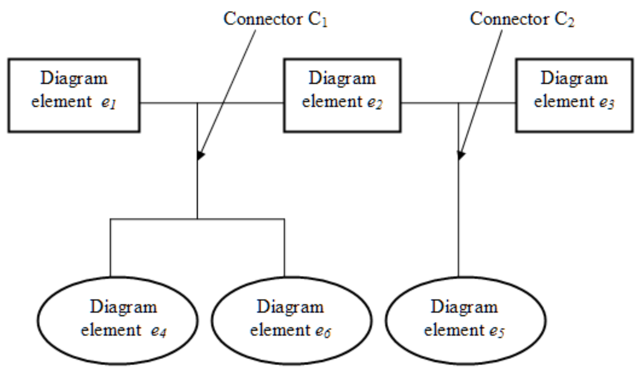

Fig. 1. Exemplary diagram consisting of two connectors sharing a common element e_2.

denotes arrow-less connector endpoint. Multisets are applied here instead of simple sets because the diagram element can be connected with itself. In such case there exist two connector endpoints associated with the same element. It leads to the appearance of the pair containing this element twice in V_C. Finally, the product of the diagram structure recognition is the set of connectors and the family of corresponding multisets of connected diagram elements:

$$(\widehat{C},\widehat{U}) = (\{C_i : i = 1,...,M\}, \{V_{C_i} : i = 1,...,M\}), \quad (1)$$

where M is the number of detected connectors. A DE may belong to more than one multiset V_{C_i} if it is connected with other DEs by various connectors. The case where the central element is shared by two connectors is shown in Fig.1. The elements of the diagram are connected by two connectors: C_1 and C_2. In this case, the structure recognition procedure builds the following diagram description:

$$\begin{aligned}(\widehat{C},\widehat{U}) = (&\{C_1, C_2\},\\ &\{\{(e_1, false), (e_2, false), (e_4, false), (e_6, false)\}, \quad (2)\\ &\{(e_2, false), (e_3, false), (e_5, false)\}\}).\end{aligned}$$

IV. Description of the method

Unfortunately, the formulation of the diagram structure recognition problem does not lead to an efficient solution, other that "brute force" approach based on exhaustive search of all subsets of L, which is obviously infeasible in most practical cases. Therefore, we propose simplified suboptimal solution that leads to construction of a connector set $\{C_1, C_2, ..., C_M\}$, which however does not guarantee that the minimal number of connectors are found. On the other hand however, it applies some intuitive principles that humans typically apply when trying to read a structure on a diagram presented on an image. Experiments described in section V show that diagram structure recognized with the proposed algorithm is close to the human interpretation of test diagram images.

The approach taken consists in the observation that when a human tries to find connectors in a diagram by sight, it intuitively starts with long edges and tries to interpret them as "simple connectors" connecting pairs of DEs. Then a human tries to find "branches" that connect other DEs to previously found simple connectors. Next, one tries to find inter-connectors, i.e. polylines that connect previously found connectors. Finally, we (humans) try to merge already found connectors into more complex ones by finding intersecting lines belonging to various connectors that are candidates for merging. It is a process that starts with simple and obvious connectors and gradually extend them to more complex structures. This process can be modeled as a procedure implemented in a computer. Each stage outlined above is in fact an iterative operation that processes successive items (edges, polylines, simpler connectors - depending on the stage), where each iteration may lead to modification of the connector set obtained so far. The modification is however conditioned on the rules set presented in the previous section. If it leads to violation of constraints defined by the rules then the modification is not carried out. In this way, at each stage of the procedure we have the diagram structure that is consistent with the assumed principles.

Now the procedure will be described in more details. The input to the procedure is the set of diagram elements E and the set of line segments (vectors) obtained from the raster image vectorization procedure (vectorizer). The applied vectorization procedure is based on the algorithm described in [14]. We will not deal here with the details of methods of shape recognition used to obtain the set E. They are briefly described in the subsection IV-I. In the proposed algorithm, the set of constants usually applied as thresholds are utilized. Values of these thresholds were estimated experimentally by analyzing a set of typical diagram images from the validation set. The selected validation set is disjoint from the testing set used in order to evaluate the method performance.

The connectors finding method consists of the following steps:

A. Edge detection

The aim of this step is to create the set of edges L from the set of line segments fetched by the vectorizer. We use the term "edge" to emphasize the difference in relation to the notion of simple line segment which is the direct product of vectorization. The line segments created by the vectorizer should not be used directly in further steps of the procedure. Our experiments showed that there are some troublesome artifacts, especially at intersections of relatively thick lines or at vertices of polylines. They are short line segments of the length of single pixels that connect longer line segments. Such geometrical structures need a kind of smoothing in order to obtain longer and straight line segments, most likely being "true" lines in the original diagram. Here we call such smoothed and merged line segments "edges". An example of an erroneous line segment structure created by the vectorizer is presented in Fig.2. The unwanted artifacts are indicated by blue circles.

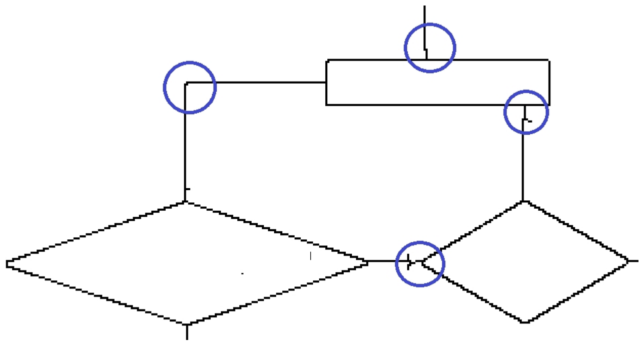
Fig. 2. Examples of inaccuracy artifacts introduced at the stage of image vectorization

The procedure takes the line segment that is not yet assigned to any element in E nor to any edge in L and tries to extend it to a longer edge by building a polyline being a sequence of interconnected line segments. In each iteration the algorithm tries to attach the next line segment that is adjacent to one of end points of the already created polyline. This segment is selected that is most collinear with the straight line approximating already created polyline. The attachment criterion is used to select candidates for extension. It takes into account the angle between the candidate segment and already approximated line and the length of the candidate. Short segments (being probably artifacts of the vectorization procedure) can be connected even if the angle is relatively big. The logical predicate used as the extension criterion is as follows:

$$\angle(e,c) < \alpha_{max} \vee len(l_t) \leq l_{min} \vee len(c) \leq 1.5 * w, \quad (3)$$

where e is the edge (single line segment) approximating the polyline created so far, c is the line segment - the candidate for extension, l_t is the terminal line segment in the polyline that is adjacent to c and $len(\bullet)$ is the length of the line segment. w is the average width of the line segments already attached to the polyline. α_{max} was experimentally set to $10°$.

According to the criterion (3), the angle between connected segments must be small enough or at least one of adjacent segments (the candidate one or the terminal segment) is short enough. This alternative makes it possible to use residual vectors of the length of 1-2 pixels that are artifacts of the vectorization procedure. Such residual vectors often are oriented at big angles with the relation to its (longer) neighbors. The procedure iteratively tries to extend the polyline until no new segments can be attached. After each extension the new linear approximation of the polyline (denoted by e in the formula 3) is evaluated by least square fitting of points on the polyline to the approximating line.

B. Preparation to processing

The aim of this stage is to identify DEs that are close enough to endpoints of edges created in the previous stage. In this way it is possible to identify edges that are candidates for terminal edges of connectors. The terminal edge of the connector is the edge directly attached to the element connected by a connector. Then for all edge vertices the closest DE is found that is within assumed maximal allowed distance from the vertex. The tolerance is estimated depending of the shape and edge line widths. By analyzing the set of exemplary diagrams we assumed that the tolerance should be evaluated as $min(3 * max(w_e, w_s), 0.25 * s_{BB})$ where w_e is the width of the edge line, w_s is the width of the line of the DE shape and s_{BB} is the smaller of x and y sizes of the bounding box enclosing the element. The angle between the closest element edge and the connector edge is also taken into account to avoid considering as very close an edge that is almost parallel to an edge of DE. As the result of this stage, each edge endpoint is annotated either with the index of the close diagram element or with "dummy" index, denoting that there is no close element to the edge endpoint. Additionally, the "edge structure" is created, that makes it possible to quickly find all edges in L adjacent to a given vertex.

C. Finding simple connectors

At this stage simple connectors are being found, where two DEs are directly connected by a single edge. It lies in finding edges with two endpoints marked with various diagram elements. Cases where the edge connects the shape with itself and is completely within this shape bounding box are excluded by applying one of constraints defined in Section III.

D. Finding polyline connectors

This stage consists in finding edge sequences (a polyline consisting of edges) that connect two shapes. The procedure used here starts with a polyline consisting of a single edge that is not yet assigned to any connector. It iteratively tries to extend the polyline by attaching its left/right endpoint neighbors until no further extension is possible or the attached edge is connected to an element. Backtracking is applied in cases where there are many neighboring edges adjacent to the terminal edge in the polyline and extending edge selection in certain iteration leads to the polyline that neither can be further extended nor it terminates with the edge adjacent to any DE.

The procedure is being repeated, each time starting with an edge that is not yet assigned to any connector. If it leads to the polyline connecting two diagram elements then it is assumed to be the polyline connector. The new connector is then created and all edges are labeled as assigned to this connector.

E. Finding inter-connectors

In the previous stages only such connectors (or fragments of connectors) were recognized which link pairs of DEs. In the next phase, new polyline connectors are tried that connect already found connectors with other connectors or diagram elements. The interconnector appears for instance in the diagram in Fig.1. At the first stage, the simple connector between elements e_1 and e_2 is found. The polyline connector linking elements e_4 and e_6 is detected in the second stage. In the current stage the interconnector that links this two simpler connectors will be recognized. The case of "branch"

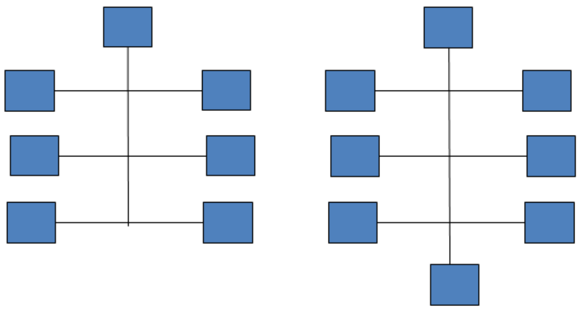

Fig. 3. Pine-like connector structures

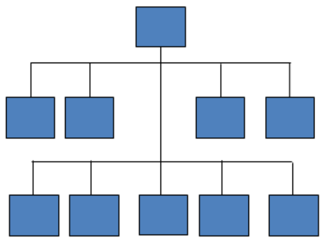

Fig. 4. Comb-like connector structures

that extends simpler connectors appears in Fig.1 in the case of the single connector that links elements e_2 and e_3. The branch consisting of the vertical edge in the right part of the diagram links the simple connector with the element e_5.

The procedure iterates until a single iteration does not result in any extension of the obtained connector sets. The single iteration in turn, consists of subiterations that iterate over all unassigned line segments, where each unassigned segment is tried to be extended into a polyline connecting two earlier detected connectors or a connector and a diagram element. Typically, this stage creates T-connections. T-connection is the connection of edges, where one edge perpendicular to another one touches them in the middle. The connection between the edge linking e_2 with e_3 and the vertical edge adjacent to e_5 is a typical T-connection.

The next two stages are aimed on merging connectors found earlier into more complex ones. Pairs of connectors that are candidates to merging must have intersecting edges. Merging all connectors having intersecting edges in many cases would lead to a structure not intended by the diagram author. Actually, the problem of simpler connector merging seems to be the hardest one in the process of diagram structure recognition. We distinguished two specific graphical configurations of edges that are typically used when composing diagrams and we perform connector merging only if the merged connector conforms to the one of these specific configurations. We called these configurations *pine-like* and *comb-like* structures. The structures are shown in Fig.3 and Fig.4

F. Constructing pine-like connectors

In this step, the specific type of connector is detected which consists of a simple single edge that intersects other connectors. It is assumed that in such case the diagram creator intention was to depict the situation where elements connected by such a connector are connected each to another.

In order to find pine-like connectors, the simple connectors (being just single edges) are tested against intersection with edges of other connectors, let us call it *trunk*. All other simple connectors that intersect the trunk are merged to the group containing the trunk.

G. Constructing comb-like connectors

The comb-like structure is presented in Fig.4. It consists of the vertical trunk connector intersected by one or more "combs". Comb is a connector with the principal horizontal edge to which a series of simple vertical edges (branches) are T-connected. Additionally, it is required that comb teeth are approximately of to the same length in the interval $< 0.1 * l_{pr}, 0.7 * l_{pr} >$ where l_{pr} is the length of the principal comb edge. The comb-like structure must also have trunk - the vertical line being a connector that have a single DE at its top and the trunk length must be at least $2 * l_{t_{min}}$, where $l_{t_{min}}$ is the shortest tooth length of the comb. The procedure of comb-like connector construction consists in finding connectors that satisfy aforementioned conditions. The set of connectors that satisfies it is replaced in the set $\{C_1, C_@, ..., C_M\}$ by the product of the merge operation.

H. Merging connectors by dot-markers

Finally, all these pairs of connectors are merged that contain lines that cross one with another, where there is a dot-marker at the intersection of lines belonging to various connectors. In order to consider two line segments as marked for merging, the following conditions must be satisfied:

- intersecting lines are approximately of the same width $(0.5 \leq w_1/w_2 \leq 2.0)$, where w_1, w_2 are widths of lines;
- lines are approximately of the same color, the color tolerance is defined for the components of 24-bit RGB color space;

$$\begin{aligned} R_{max} - R_{min} &< 30, \\ G_{max} - G_{min} &< 30, \\ B_{max} - B_{min} &< 30, \end{aligned} \tag{4}$$

- the angle between intersecting lines is approximately the right angle (with the tolerance range from 80° to 100°);
- the four diagonal pixels at the distance $\sqrt(2) * max(w_1, w_2)$ from the lines intersection point are closer in colors to the average lines color than to the background color.

The last condition is responsible for detecting the dot intersection marker at the lines intersection point.

I. Diagram elements recognition

Because this paper is mainly focused on the recognition of connections between diagram elements, we will only briefly describe methods used in order to recognize diagram elements. We assume that diagram elements are: a) polygons, b) circles, ellipses or arches of ellipses and c) shapes being combination of a) and b), e.g. the symbol of a drum often used in flowcharts to denote mass storage. Methods used to recognize polygonal shapes are based on rules that define the mutual geometrical relations between line segments constituting polygon edges. For example, the parallelogram not being just a rectangle is defined as the sequence of 4 edges (l_0, l_1, l_2, l_3) defined by their endpoints $(p_i^{(B)}, e_i^{(E)}), i = 0, ..., 3)$ that satisfy the set of constraints:

- $\mid p_i^{(E)} - p_{i\oplus 1}^{(B)} \mid \leq \epsilon$ for $i = 0, ..., 3$;
- either $p_i^{(E)} = p_{i\oplus 1}^{(B)}$ or $p_i^{(E)}$ is connected with $p_{i\oplus 1}^{(B)}$ by a chain of short line segments that are entirely included inside the bounding box defined by $p_i^{(E)}$ and $p_{i\oplus 1}^{(B)}$;
- $l_0 \parallel l_2$ and $l_1 \parallel l_3$;
- $\angle(l_0, l_1) \leq 90° - \alpha_{toll}$ or $\angle(l_0, l_1) \geq 90° + \alpha_{toll}$,

where $i \oplus 1 = (i + 1) \mod 4$ and α_{toll} denotes the tolerance for right angles. The last constraint makes it possible to distinguish between rectangles and other parallelograms.

The procedure of shape recognition starts with a line segment from the vector set created by the vectorization procedure and successively tries to extend it to a sequence of segments, so that the constraints defined for allowed shapes are satisfied. It may happen that certain sequence of line segments satisfies constraints for more that single shape. For example, due to drawing inaccuracies, a quadrilateral found in the diagram may satisfy both constraints for the rectangle, the trapezoid as well as for the rounded vertices rectangle. In such a case, the measure of inaccuracy for all candidate shapes is computed and this shape is finally selected for which the inaccuracy measure is lowest. The procedure is repeated, each time starting with a next line segments that is not already assigned to any shape, until all unassigned line segments are tried.

In the case of ellipses and arches the procedure starts with a candidate edge and tries to extend it to a polyline that best approximates the ellipse fragment. Only axis-aligned ellipses are considered. Let $S = (l_1, l_2, ..., l_n)$ be a sequence of line segments that approximate an ellipse arch. At each stage the procedure tries to extend it with one of line segments that is adjacent to l_1 or l_n. Such extending segment is selected for which the ellipse approximation error is the lowest. The approximation error is the average distance of pixels constituting the polyline S to the best fitting ellipse. The pixel set used for approximation is created by applying Bresenham line drawing algorithm to all lines in the set S. The axis-aligned ellipse is defined by four parameters: coordinates of the ellipse center (x_c, y_c), the length of x-axis d_x and the shape factor a - the ratio of x and y axes lengths $a = d_x/d_y$. The best fitting ellipse is found using the method described in [15]. The parameters of the optimal ellipse as assumed to be within the reasonable ranges determined in relation to the image size, e.g. the ellipse (x_c, y_c) center must be within the range defined by the image resolution and both ellipse axes must be not longer than the corresponding image size along x or y axes. If the parameters computed by the optimization procedure are out of these ranges then the approximation is assumed to fail and another extending line is tried. The extension is continued until the closed ellipse is obtained or no more lines can be attached. The final ellipse arch is accepted if it constitutes at least 50% of the complete ellipse. This acceptance threshold may seem to be high, but shapes that we consider here as diagram elements never consist of shorter ellipse fragments. On the other hand, setting too low value of the threshold may lead to false recognition of other shape elements as ellipse fragments.

V. EXPERIMENTS

A. Evaluation of diagram structure recognition accuracy

The accuracy of diagram structure recognition can be assessed by the complexity of operations necessary to convert the recognized structure into the correct one (ground truth). This complexity can be measured by the summed cost of elementary operations that can be used in order to turn the recognized structure into the correct one. We focus here on the evaluation of the multiset $\widehat{U}$ as defined in (1). Let us assume that the recognized structure described by $\widehat{U}$ is to be converted into the correct structure $\widehat{U}^*$ by applying the sequence of elementary operations. In the result, the sequence of structures is created: $(\widehat{U} = \widehat{U}_1, \widehat{U}_2, ..., \widehat{U}_K = \widehat{U}^*)$ where $\widehat{U}_k$ is converted into $\widehat{U}_{k+1}$ by applying one of the following elementary operations from the set $\mathfrak{O} = \{o_C, o_S, o_M, o_C, o_E, o_R, o_A, o_D\}$ defined as follows:

- o_C - creating a new connector that links two DEs;
- o_S - splitting the multiset $U_i \in \widehat{U}_k$ into two multisets $U_i^1, U_i^2 \in \widehat{U}_{k+1}$ - it corresponds to dividing the compound connector into two simpler ones;
- o_M - merging two multiset $U_i, U_j \in \widehat{U}_k$ into the single multiset $U_l \in \widehat{U}_{k+1}$ - it corresponds to merging two connectors;
- o_E - adding a branch to a diagram element to a connector, i.e. replacing $U_i, \in \widehat{U}_k$ by the new multiset $U_i' = (U_i \cup (e, t)) \in \widehat{U}_{k+1}$;
- o_R - removing a branch to a diagram element from a connector, i.e. replacing $U_i, \in \widehat{U}_k$ by the new multiset $U_i' = (U_i \setminus (e, t)) \in \widehat{U}_{k+1}$;
- o_A - changing the arrow status of the connector endpoint for a certain element descriptor (e, t) in a certain multiset $U_i, \in \widehat{U}_k$.
- o_D - discarding of the whole result of recognition, i.e. replacing of $\widehat{U}_1$ by the empty set.

The last operation is only allowed as the first one in the conversion sequence and can be applied if the recognition result is extremely different from the ground truth diagram structure. Each operation has its cost. The accuracy of the recognized diagram structure can be assessed by the total cost of the least costly operation sequence that converts $\widehat{U}$ into $\widehat{U}^*$. This concept is similar to the edit distance widely used e.g. in automatic speech recognition accuracy evaluation or spelling errors correction ([16]). The value computed in this way is however the absolute measure of labor amount necessary to make a correction to the recognized structure. Certain value of the edit distance may indicate quite good accuracy in the case of very complex diagram, while it may correspond to a poor accuracy in the case where the diagram consists just of few elements and connector edges. Therefore the relative measure related to the actual diagram complexity seems to be more appropriate. The diagram complexity can be measured by the cost of operations necessary to build the set $\widehat{U}^*$ from very beginning, i.e. from the empty family of multisets, using only elementary operations. Let p_o denote the cost of the operation o, and $n_o^{(\widehat{U}_1)}$ and $n_o^{(\emptyset)}$ denote the counts of the operation o that must be applied to obtain the correct diagram description $\widehat{U}^*$ from the actual recognition result $\widehat{U}_1$ and from the empty set correspondingly. The total costs of corrections and building from the very beginning are:

$$\begin{aligned} P_{corr} &= \sum_{o \in \mathfrak{O}} p_o * n_o^{(\widehat{U}_1)}, \\ P_{build} &= p_D + \sum_{o \in \mathfrak{O}} p_o * n_o^{(\emptyset)}. \end{aligned} \tag{5}$$

and the final recognition accuracy can be computed as:

$$Q = \frac{P_{build} - P_{corr}}{P_{build}} \in < 0, 1 > . \tag{6}$$

The value of Q is normalized into $< 0, 1 >$ interval. If the diagram is perfectly recognized then $P_{corr} = 0$ and $Q = 1$. On the other hand, if the recognition procedure totally misses then the cost of correction is not higher that discarding all recognition results (operation o_D) and building the structure from the very beginning. In this case $P_{corr} = P_{build}$ and in result $Q = 0$.

B. Hardware environment and efficiency issues

The described algorithm was implemented in C++ language. Tests were carried out using the PC equipped with Intel i7 3610QM CPU and 16GB of RAM. The execution time of the algorithm varies depending on the image contents. For simple diagrams consisting of just a few DEs connected by simple connectors the algorithm is executed in less than a single second. The longest execution time (17.6 sec.) was observed in the case of the complex diagram consisting of 58 DEs. The average processing time (including image vectorization) of a single diagram was 2.4 sec. Currently the algorithm implementation is fully sequential.

C. Results

The accuracy of diagram structure recognition was tested using three types of diagrams that differ in the complexity of intersecting connectors: a) flowcharts, b) organizational charts and c) digital circuit block diagrams. The test set consisted of 11 digital circuit diagrams, 15 flowcharts and 17 organizational charts. Flowcharts seem to have simplest connector structures, while in the case of digital circuits the intersecting connectors appear very often. Hence, the later type of diagrams is the most difficult to recognize. Because in this article we are dealing merely with the problem of connectors recognition, we selected for the tests only such diagrams, where there were no mistakes in diagram elements recognition. For each recognized diagram structure the counts of operations from the set $\mathfrak{O}$ that are necessary to correct the structure were determined as well as the number of operations necessary to build the diagram from the very beginning. We assumed that in the process of construction of the diagram only the operations of new connector creation (o_C) and extension of the existing connector with an additional branch (o_E) were used. Finally the recognition quality Q was computed for each diagram. We assumed that the unit cost p_o of each operation o is equal to 1.0. The results are presented in Table I. Columns in the left part of the table include numbers of individual correcting operations from the set $\mathfrak{O}$, summed by types of diagrams. The meaning of operation symbols used in Table I were explained in the previous section. The column containing costs of correction/creation contain average costs for individual types of diagrams. The bottom row presents the results analogous to described for individual types of diagrams, but now they are prepared for the whole set containing all types of diagrams.

The small number of diagrams used in the tests do not give rights to create very general conclusions, although the average result for all diagrams equal to 92% is very promising. From the perspective of diagram classes, the worst average result (88% - which seems to be pretty good) was obtained for the class of digital block diagrams. This set contains the most difficult intersecting connections. Our subjective evaluation of the obtained results is very optimistic. Most errors were caused by low image quality and inaccuracies in drawing. It can be observed that there were no errors consisting in detecting neither "false" (i.e. actually not existing) connectors nor branches. The most typical error consisted in omitting connector branches to some DEs. Detailed analysis of missing branches revealed that in most cases errors of this kind were caused by too wide gaps between a diagram element and a terminal endpoint of the connector branch.

Table II presents the results of diagram structure recognition of the exemplary organizational diagram. Fig. II.1 shows the original image. All detected connectors are drawn in Fig. II.2. The connectors are drawn with thick blue lines. Remaining figures show selected individual connectors detected in this diagram. Some simple connectors were omitted and only more complex ones are presented. It is clear that intersecting connectors were properly separated and complex connectors with "branches" were constructed as intended by the diagram author. In the case of this diagram all connectors were recognized correctly.

VI. Conclusion

The subjective and objective evaluation of the method gives us good perspective for further development of the method,

TABLE I
STRUCTURE RECOGNITION ACCURACY EVALUATION ON THREE TYPES OF DIAGRAMS

Diagram type	**Correction (number of correcting operations)**						**Construction (number of constructing opeartions)**			**Q**
	o_C	o_S	o_M	o_E	o_R	Correction cost	o_C	o_E	Construction cost	
Digital circuits	3	0	8	0	19	30	191	22	224	0.88
Flowcharts	2	0	1	0	10	13	178	12	205	0.94
Organizational diagrams	2	0	1	0	12	15	308	14	339	0.96
Total:	7	0	10	0	41	58	677	48	768	0.92

although we realize that the set of diagrams used in order to asses the quality of structure recognition is quite small and conclusions drawn from described experiments may be disturbed by the random factor. In future, significantly bigger set of diagrams will be manually annotated and used for reliable accuracy estimation.

The main weakness of the proposed method seems that it utilizes the set of constants-thresholds, in most cases defining tolerances of various graphical attributes (lengths, angles, widths, distances etc.). It makes possible to correctly recognize some structures that are drawn inaccurately but, on the other hand, it may also lead to recognition errors. In our experiments we tuned these constants intuitively, so as to obtain best recognition results on the validation image set (which is disjoint from the test set). We cannot however assure that these parameter values are selected optimally. In future works, other method of parameter tuning should be applied that will make it possible to find suboptimal parameter values without the engagement of a human. Image processing methods used at the stage of diagram image binarization and vectorization should also be improved, so that low quality images as well as images with background patters or containing "decorated" shapes can be more reliably analyzed.

Some problems encountered in determining connections between diagram elements are caused merely by low quality of scanned images, while others follow from complicated shapes of connectors that intersect each other. In future, an experiment will be carried out which will compare the average accuracy achieved in a set of scanned images with the accuracy achieved in a set of images directly converted to the raster format, e.g. by saving the image created in a graphic diagram editor.

ACKNOWLEDGMENT

This work was partially supported by the European Commission under the 7th Framework Programme, Coordination and Support Action, Grant Agreement Number 316097, ENGINE - European research centre of Network intelliGence for INnovation Enhancement.

REFERENCES

[1] R. Datta, D. Joshi, J. Li, and J. Wang, “Image retrieval: Ideas, influences, and trends of the new age,” *ACM Computing Surveys*, vol. 40, no. 12, pp. 5:1–5:60, 2012. doi: 10.1145/1348246.1348248. [Online]. Available: http://doi.acm.org/10.1145/

[2] D. Blostein, “General diagram-recognition methodologies,” *Graphics Recognition Methods and Applications*, vol. 1072, pp. 106–122, 1996. doi: 10.1007/3-540-61226-2-10. [Online]. Available: http://dx.doi.org/10.1007/3-540-61226-2_10

[3] Y. Liu, X. Lu, Y. Qin, Z. Tang, and J. Xu, “Review of chart recognition in document images,” in *Proc. SPIE 8654, Visualization and Data Analysis*, vol. 865410, 2013. doi: 10.1117/12.2008467. [Online]. Available: http://dx.doi.org/10.1117/12.2008467

[4] A. Lemaitre, H. Mouch‘ere, J. Camillerapp, and B. Couasnon, “Interest of syntactic knowledge for on-line cognitionr,” in *Proc. of ninth IAPR International Workshop on Graphics Recognition (GREG2011)*, 2011. doi: 10.1007/978-3-642-36824-0 9 pp. 85–98.

[5] M. Bresler, D. Prusa, and V. Hlavac, “Modeling flowchart structure recognition as a max-sum problem,” in *ICDAR, IEEE Computer Society*, 2013. doi: 10.1109/ICDAR.2013.246 pp. 1215–1219. [Online]. Available: http://dblp.uni-trier.de/db/conf/icdar/icdar2013.html/BreslerPH13

[6] G. Feng, C. Viard-Gaudin, and Z. Sun, “On-line hand-drawn electric circuit diagram recognition using 2d dynamic programming,” in *Pattern Recognition*, vol. 42, 2009. doi: 10.1016/j.patcog.2009.01.031 pp. 3215–3223. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00419076

[7] Y. Qi, M. Szummer, and T. P. Minka, “Diagram structure recognition by bayesian conditional random fields,” in *CVPR 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 191 – 196.

[8] *Evaluating Flowchart Recognition for Patent Retrieval*, 2013. doi: 10.1007/s10791-013-9234-3. [Online]. Available: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/EVIA/08-EVIA2013-LupuM.pdf

[9] A. Hanbury, N. Bhatti, M. Lupu, and R. Mörzinger, “Patent image retrieval: a survey,” in *Proceedings of the 4th worshop on Patent information retrieval*, 2011. doi: 10.1145/2064975.2064979 pp. 494–497.

[10] L. Yan, W. Huang, and C. L. Tan, “Semi-automatic ground truth generation for chart image recognition,” in *Workshop on Document Analysis Systems (DAS)*, 2006. doi: 10.1016/j.patrec.2015.02.001 pp. 324–335.

[11] M. Awadalla and A. Sadek, “Spiking neural network-based control chart pattern recognition,” *Journal of Engineering and Technology Research*, vol. 3, no. 1, pp. 5–15, 2011. doi: 10.1007/s10845-012-0659-0. [Online]. Available: http://www.academicjournals.org/journal/JETR/article-abstract/59E571210699

[12] Y. Yu, A. Samal, and S. C. Seth, “A system for recognizing a large class of engineering drawings,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1997. doi: 10.1109/34.608290

[13] J. Sas and A. Zolnierek, “Three-stage method of text region extraction from diagram raster images,” in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013, Milkow, Poland, 27-29 May 2013*, 2013. doi: 10.1007/978-3-319-00969-8-52 pp. 527–538. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-00969-8_52

[14] A. N. Kolesnikov, V. V. Belekhov, and I. O. Chalenko, “Vectorization of raster images,” *Pattern Recognition and Image Analysis*, vol. 6, no. 4, pp. 786–194, 1995. [Online]. Available: http://cs.joensuu.fi/~koles/dissertation/Kolesnikov_Paper1.pdf

[15] A. Fitzgibbon, M. Pilu, and R. B. Fisher, “Direct least square fitting of ellipses,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, May 1999. doi: 10.1109/34.765658. [Online]. Available: http://dx.doi.org/10.1109/34.765658

[16] R. A. Wagner and M. J. Fischer, “The string-to-string correction problem,” *J. ACM*, vol. 21, no. 1, pp. 168–173, Jan. 1974. doi: 10.1145/321796.321811. [Online]. Available: http://doi.acm.org/10.1145/321796.321811

TABLE II
EXAMPLE OF RECOGNIZED DIAGRAM STRUCTURE

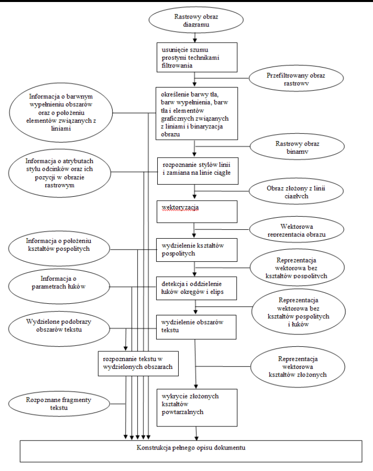	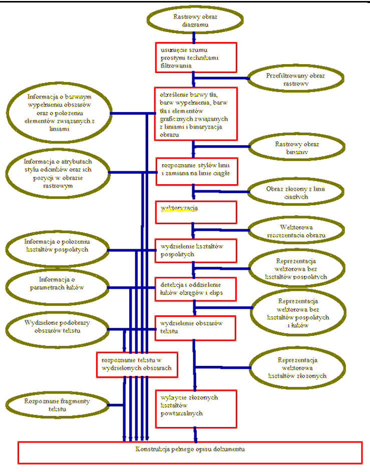	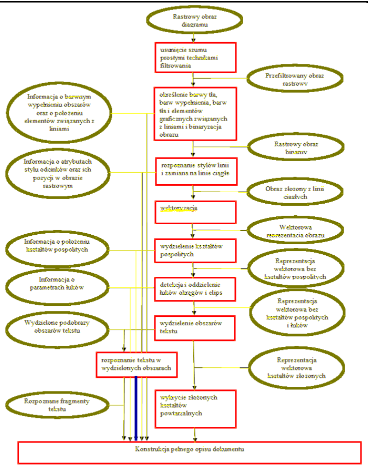
Fig.II.1. Original diagram	Fig.II.2. All detected connectors	Fig.II.3. Example of simple connector
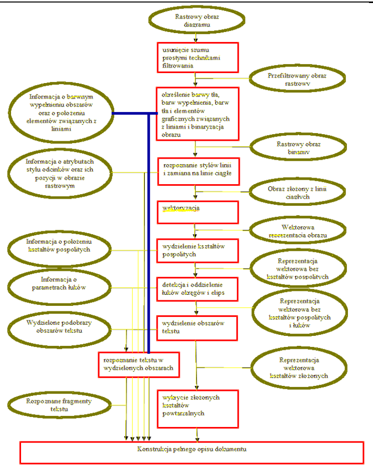	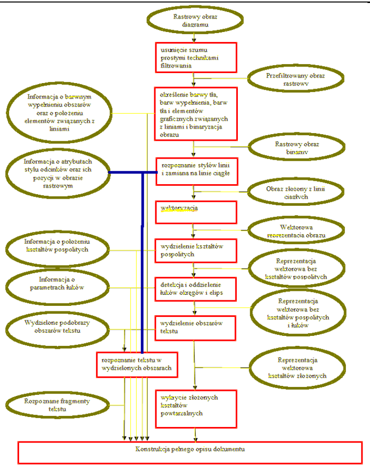	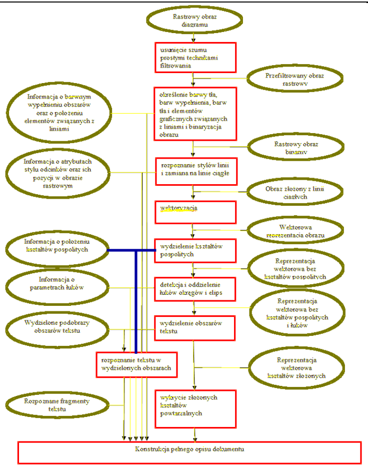
Fig.II.4. Example of compound connector	Fig.II.5. Example of compound connector	Fig.II.6. Example of compound connector
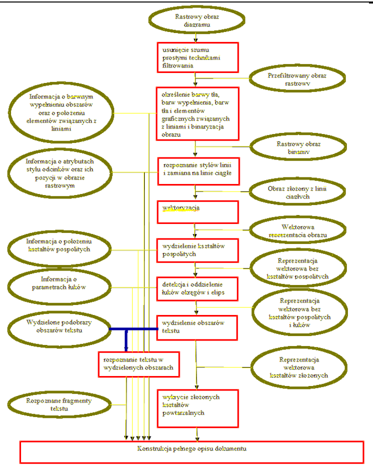	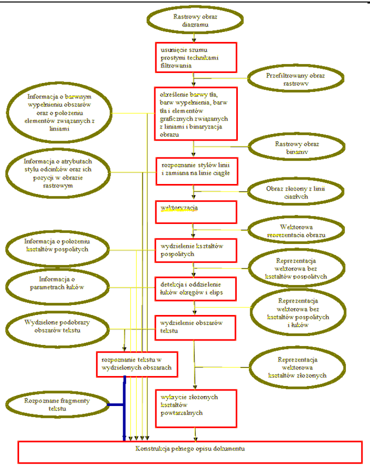	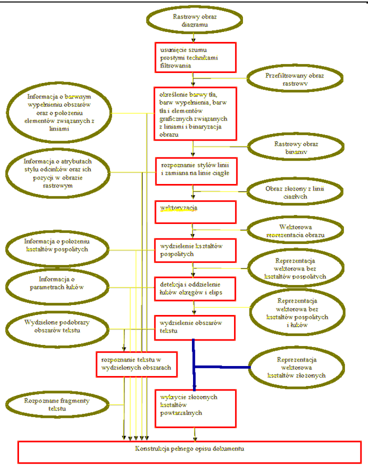
Fig.II.7. Example of compound connector	Fig.II.8. Example of compound connector	Fig.II.9. Example of compound connector