A Performance Evaluation Protocol for Graphics Recognition Systems

Ihsin T. Phillips[1], Jisheng Liang[2], Atul K. Chhabra[3], and Robert Haralick[2]

[1] Department of Computer Science/Software Engineering, Seattle University, Seattle, Washington 98122

[2] Department of Electrical Engineering, University of Washington, Seattle, Washington 98195

[3] Bell Atlantic Network Systems, Advanced Technology, 500 Westchester Avenue, White Plains, NY 10604, USA

Abstract. This paper defines a computational protocol for evaluating the performance of raster to vector conversion systems. The graphical entities handled by this protocol are continuous and dashed lines, arcs, and circles, and text regions. The protocol allows matches of the type one-to-one, one-to-many, and many-to-one between the ground truth and the recognition results.

1 Introduction

Systems which convert existing paper-based drawings into electronic format are in demand and a few have been developed. However, the performance of the prototypes and commercial systems is either unknown, or only reported in a limited way by the system developers. An evaluation for these systems, or their subsystems, would contribute to the advancement of the field. Responding to this need, a dashed-line detection competition for developers of dashed-line detection algorithms was proposed and took place during the first International Workshop on Graphics Recognition at Penn State University, in 1995. A benchmark[1] was developed and used in that competition. That benchmark includes a performance evaluator and a software tool that automatically generates dashed-line test images and the corresponding groundtruth.

In this paper, we extend that protocol to evaluate the performance of graphics recognition systems on images that contain straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), and text blocks. (Engineering drawings primarily use a combination of these geometric elements. Therefore, despite being restricted to these simple entity types, the evaluation protocol is applicable to a wide variety of drawings. Upgrading the evaluator to handle other types of entities is straight forward. To do this, one needs to provide the evaluator the parameters of the entity or entities and the performance evaluation criteria.) To evaluate a given recognition system, the system is tested on a set of pre-selected test images. (The groundtruth for the test images must be reliable.) The results of the recognition system are matched, using the criteria defined in this protocol, with the corresponding groundtruth of the test images. The matching results are the numbers of one-to-one matches, one-to-many matches, many-to-one matches, as well as the numbers of false-alarms and misses. Performance measurements for the recognition system can be formulated using a linear combination of some or all of the matching results.

Our evaluator is designed to be used by recognition system researchers and developers for testing and enhancing their recognition algorithms. The evaluator allows the users to select 'text-only', 'graphics-only', or 'all' option for their systems performance evaluations. The 'text-only' evaluation option is designed for recognition systems that detect only the text blocks in the input image. The 'graphics-only' evaluation option is designed for recognition systems that detect only the graphical entities. The 'all' evaluation option is designed for systems that can detect both graphics and text blocks.

This paper is organized as follows: In Sect. 2, we give a brief review of some related work. In Sect. 3 we specify the parameters for the entities. The protocols for performance evaluation and entity matching are given in Sect. 4. In Sect. 5, we present the matching criteria for each pair of valid combinations. The precise definitions of the line-to-line matching functions are given in Appendix A.

2 Previous Work

Performance evaluation and benchmarking have been gaining acceptance in all areas of computer vision. An overview of this area is available at [2]. Performance evaluation of graphics recognition is still a very young field; objective and quantitative methods for evaluation of graphics recognition have been proposed very recently [1, 3, 4, 5]. Kong et al. [1] propose a quantitative method for evaluating the recognition of dashed lines. Hori and Doermann [3] propose a quantitative performance measurement methodology for task-specific raster to vector conversion. Wenyin and Dori [4] present a protocol for evaluating the recognition of straight and circular lines. All of these methods are limited in their applicability.

Kong et al. [1] use angle, distance, relative overlap, and offset between line segments for evaluating line matches and for detecting line styles. They use several arbitrary and rigid thresholds. They do not allow for fragmentation of detected lines.

Hori and Doermann [3] instantiate and extend Haralick's framework for performance characterization in image analysis [6], in an application-dependent manner, for measuring the performance of raster to vector conversion algorithms. The 'applications' addressed in the work are thinning, medial line finding, and line fitting – all low level techniques that do not completely constitute vectorization. It is hard to extend the work to evaluate a complete vectorization system. Hori and Doermann's protocol does not distinguish between detection rate and false alarm rate. It does not include an overall evaluation metric. It does not allow for fragmentation of detected lines.

Wenyin and Dori [4] propose performance evaluation indices for straight and circular line detection. Detection and false alarm rates are defined at both the pixel level and the vector level. Pixel level performance indices (measures of shape preservation) are not appropriate when dealing with real images that contain severe distortion introduced by warping and other defects in the hard copy drawing and by the scanning/imaging system. Attempts to obtain a high pixel recovery index would unnecessarily require the detected vectors to be true to the distorted shape of the imaged lines, thereby making the detected lines fragmented. Wenyin and Dori weight all true positives and false positives by their respective lengths. This is inappropriate if the goal of the evaluation is to measure the cost of post-processing operations that are necessary to correct the mistakes of vectorization. The time for manual post processing does not depend significantly on the length of a true positive, a missed entity, or a false positive. Time taken for adding or deleting a line in a CAD tool does not depend significantly on the length of the line.

Neither of the above methods addresses the extraction or separation of text from graphics. It is not possible to evaluate graphics recognition systems on realistic drawings without accounting for text in the drawings. Wenyin and Dori [5] propose a protocol for evaluating text-graphics separation. In this protocol, the quality of the recognized text boxes is measured using Q_b, the basic quality, and Q_{fr}, the fragmentation quality. The protocol does not explicitly penalize overlapping text boxes; they are penalized in an indirect way. Q_{fr} implicitly penalizes overlap among recognized text boxes. For N recognized text boxes that are identical and have a 100% overlap with a text box in the groundtruth, Q_{fr} would be $1/\sqrt{N}$. The theoretical basis for penalizing overlapping text box recognition by $1/\sqrt{N}$ is not stated. Moreover, the protocol of [5] does not allow one to accept one of the N identical text boxes as a good match and to label others as false alarms. Therefore, this penalty term cannot be used to measure post-processing/editing cost.

3 Entity Definition

3.1 Entity Specification

Currently our evaluator handles seven types of entities: solid and dashed lines, solid and dashed arcs, solid and dashed circles, and text areas. The specification of the parameters for these seven types are given below.

- Solid or dashed line type: For a solid or dashed line segment, the parameters are: the entity type indicator (a solid line or a dashed line), the x- and y-coordinates (this is equivalent to the c- and r-coordinate system of the image coordinate system given in Appendix A) of the two end points (no special ordering for the two points) and the orientation of the line. The orientation we use here is in degree, clock-wise with respect to x-axis. (See Fig. 1.)
- Solid or dashed circle type: For a solid or dashed circle, the parameters are: the entity type indicator (a solid or a dashed circle), the x- and y- coordinates of the center, the radius, and the thickness of the circle arc.
- Solid or dashed arc (partial circle) type: For a solid or dashed arc, the parameters are: the entity type indicator (a solid or a dashed arc), the x- and y- coordinates of the center, the radius, the beginning and the ending angles

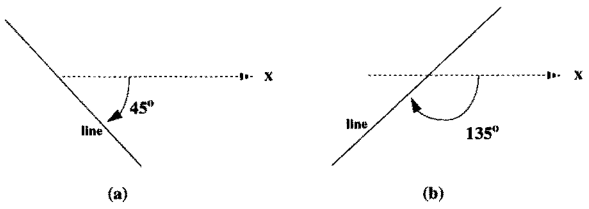


Fig. 1. The orientation of the line in (a) is 45 degrees, in (b) is 135 degrees.

(in a clock-wise order, in degrees) of the arc. The orientation of the beginning and ending angles for arcs are also in degrees, clock-wise with respect to x-axis. In some cases, the beginning angles can be larger than the ending angles. (See Fig. 2.)

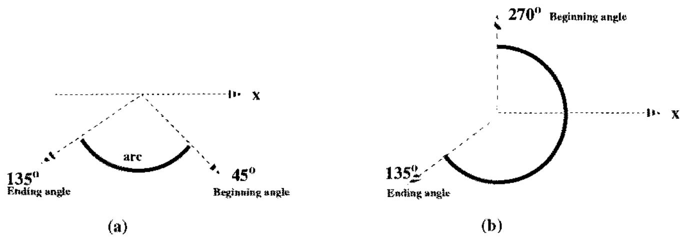


Fig. 2. The beginning and ending angles of two arcs. In (b), the beginning angle of the arc is larger than the ending angle of the arc.

- Text area type: A text area is represented by a rectangular box and its orientation. The parameters are: the entity type indicator (a text area), the x- and y- coordinates of any two opposite corners of the rectangle, and the orientation of the longest side of the rectangle.

3.2 Valid Entity Type Combinations

To speed up the matching score computation, we compute only those pairs having potential matches (e.g., line with line, etc.). The matching score for all incomparable combinations are set to zero. The following is the list of the valid entity type combinations. Others combinations are considered as incomparable.

- Solid-line with solid-line
- Solid-circle with solid-circle
- Solid-arc with solid-arc
- Dashed-line with dashed-line
- Dashed-circle with dashed-circle
- Dashed-arc with dashed-arc
- Solid-line with solid-arc

- Dashed-line with dashed-arc
- Solid-arc with solid-circle
- Dashed-arc with dashed-circle
- Text-area with text-area

4 Performance Evaluation Protocol

4.1 Evaluation Overview

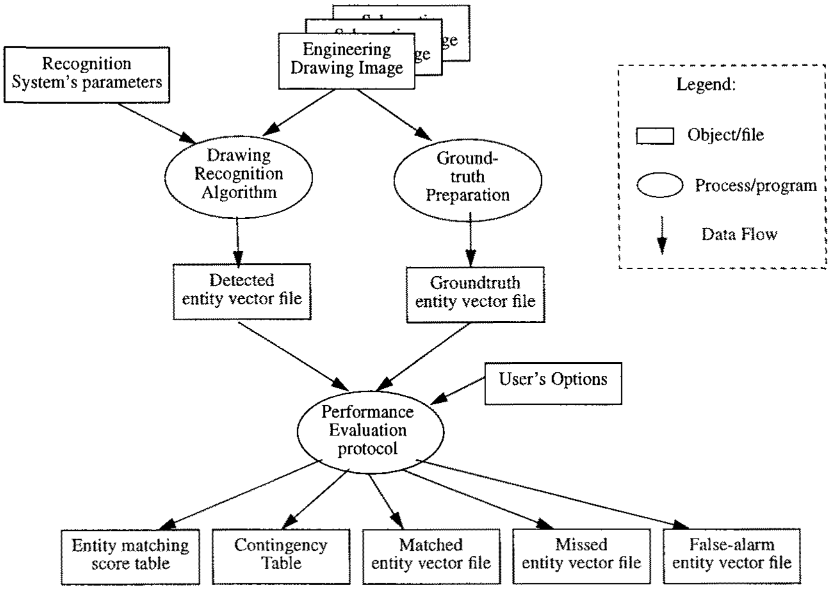

Recognition System's parameters
Engineering Drawing Image
Legend:
Object/file
Process/program
Data Flow
Drawing Recognition Algorithm
Ground-truth Preparation
Detected entity vector file
Groundtruth entity vector file
User's Options
Performance Evaluation protocol
Entity matching score table
Contingency Table
Matched entity vector file
Missed entity vector file
False-alarm entity vector file

Fig. 3. The object-process diagram of our evaluator

Inputs to the evaluator are entities (in ASCII vector format) of the recognition algorithm's output and the corresponding *groundtruth.* (Figure 3 shows an object-process diagram of our evaluator.) Since there are seven types of entities (solid and dashed lines, solid and dashed arcs, solid and dashed circles, and text areas) that are allowed, the evaluation protocol and the matching criteria are designed differently for each of the combinations. The matching scores for each pair is computed according to the pair's entity type combination, using the matching criteria defined for this combination. (These criteria are defined in Sect. 5.) A match-score table is produced from the matching score computation.

From the computed match score table, we search for all the one-to-one matches, resolving the problems of the one-to-many matches and the many-to-one matches, as well as, those false-alarms and misses. The matching results are the numbers of one-to-one matches, one-to-many matches, many-to-one matches, as well as the numbers of false-alarms and misses. Performance measurements for the recognition system can be formulated, using a linear combination of some or all of the matching results, which when weighted by application specific weights can be summed to produce an overall score relevant to the application.

4.2 Evaluation Protocol

The performance (accuracy) of a detection algorithm can be measured by counting the number of matches between the entities detected by the algorithm and the entities in the groundtruth, and the numbers of misses and false alarms. We consider a perfect result of a detection algorithm, if each and every one of the entities in the detected list matches one and only one entity of the same type in the groundtruth list and vice versa. The following is the protocol of this computation.

Step 1: Obtain the detected entities and the entities' parameters and form a detected entities list (D-list) for the entities and the parameters.
Step 2: Obtain the groundtruth entities and the entities' parameters and form a groundtruth entities list (G-list) for the entities and the parameters.
Step 3: Compute the matching score table. (See Sect. 4.3).
Step 4: Compute the one-to-one matches, resolving the problems of several matches, either from the detections or from the groundtruth. (See Sect. 4.4).
Step 5: Compute the one-to-many and many-to-one partial matches.
Step 6: Compute the false-alarms and the misses. (See Sect. 4.6).

4.3 Matching Score Table Computation

The matching score table is computed as follows. We compute the matching scores (ranging from 0 to 1, 1 being a perfect match) for each pair (with a valid combination) of entities, one from D-list (detected entities) and one from G-list (groundtruth entities), using the matching criteria defined (in Sect. 5) for the pair's combination. The matching score for all invalid combinations are set to zero. A two-dimensional data structure, the *match-score table*, is used to store the results of this computation. A higher matching score indicates a higher degree of match between the corresponding pair of entities. Figure 4 illustrates such a table. Blanks are read as zeros.

Note that entries in each row i of the match-score table represent the matching results from the i-th entity in the D-list to all entities in the G-list. Within a given row, a single entry having a high score value indicates a potential good match of the pair corresponding to that entry.

4.4 Computing One-to-One Matches

The protocol for computing the entity one-to-one matches is as follows.

Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

Fig. 4. An example of a match-score table

Step 1: We compute a two-dimensional match-count table from the computed match-score table. The entry match-count(i, j) is set to 1 if the match-score(i, j) is greater or equal to *upper-threshold*, otherwise, it is set to zero. Currently, the *upper-threshold* is set to .85. However, we allow users to set their own threshold. Figure 5 illustrates the match-count table (on the right) which is computed from the match-score table on the left.

Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

$\longrightarrow$

Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							1			
d2				1						
d3						1				
d4					1		1			
d5			1						1	
d6		1								
d7			1						1	
d8		1								

Fig. 5. An example of a match-count table (on the right)

Step 2: Two projection profiles (D-profile and G-profile) are computed from the match-count table, the result of Step 1.
The entry D(i) is computed as the sum of the matches in the i-th row of the match-count table. Likewise, the entry G(j) is computed as the sum of the matches in the j-th column of the match-count table. Figure 6 illustrates the D-profile and the G-profile computed from the match-count table.
The interpretation of these two profiles can be as follows:

- One-to-one matches: An i-th D entity has a one-to-one match with a j-th G entity, if
  - D(i) is a one,
  - the match-count(i, j) is one,
  - and G(j) is one.

Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	D-profile
d1							1				1
d2				1							1
d3						1					1
d4					1		1				2
d5			1						1		2
d6		1									1
d7			1						1		2
d8		1									1
G-profile	0	2	2	1	1	1	2	0	2	0	

Fig. 6. The two projection profiles for the match-count table of Fig. 5

(If the detection algorithm produces a perfect result, all entries in the D-profile and the G-profile will be one.)

- Many-to-one conflicts: An entry in G-profile, say G(j), is greater than one. That is, there are multiple D entities matching with the j-th entity is G-list.
- One-to-many conflicts: An entry in D-profile, say D(i), is greater than one. That is, the i-th D entity matches two or more entities in the G-list.
- False-alarms: A zero entry in the D-profile indicates that no strong match is found from this D entity to any of the entities in the G-list.
- misses: A zero entry in the G-profile indicates that no strong match is found from this G entity to any of the entities in the D-list.

Step 3: Compute the one-to-one match list.

For each D(i) that is equal to one, we attempt to locate the pair (i, j) such that both G(j) and match-count(i, j) are one, a one-to-one match (d2 and d3 in Fig. 7). We put the pair, (i, j), in the one-to-one match list, and set D(i) and G(j) to -1 (block the two entities from further consideration). Figure 7 illustrates the result of one-to-one matching. There are two such pairs.

Match-Score Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10
d1							.85		.14	
d2				1.0						
d3			.1			.9		.1		
d4					.95		.9			
d5	.25	.3	.86					.3	.88	
d6		1.0								
d7		.06	.91						.93	
d8		.91								

Match-count Table

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	D-profile
d1							1				1
d2				①							$\cancel{1}$ -1
d3						①					$\cancel{1}$ -1
d4					1		1				2
d5			1						1		2
d6		1									1
d7			1						1		2
d8		1									1
G-profile	0	2	2	$\cancel{1}$ -1	1	$\cancel{1}$ -1	2	0	2	0	

Fig. 7. One-to-one matching (the resulting one-to-one matches are circled)

Step 4: Resolving the many-to-one conflicts.

For each D(i) that is equal to one (but did not produce a one-to-one match in Step 3), we locate the pair (i, j) such that match-count(i, j) is one, but G(j) is greater than one. (There are three such pairs in Fig. 7): (d1, g7), (d6, g2), and (d8, g2).)

Without lost of generality, let D(i) and D(k) be one and let G(j) be two (meaning that there are two D entities, say i-th and k-th, matching with the j-th entity in G-list, such as (d6, g2), and (d8, g2) in Fig. 7),

Case 1: We select the pair (i, j) if match-score(i, j) >= match-score(k, j). And

- we put the pair (i, j) in the one-to-one match list;
- we set D(i) and G(j) to -1, and
- we decrease D(k) by one.

For example, if i = 6, we would select the pair (d6, g2) over (d8, g2) in Fig. 7.

Case 2: We select the pair (k, j) if match-score(k, j) > match-score(i, j) and D(k) is a one. And,

- we put the pair (k, j) in the one-to-one match list;
- we set D(k) and G(j) to -1, and
- we decrease D(i) by one.

The example, if i = 8, we still select the pair (d6, g2) over (d8, g2) in Fig. 7.

Case 3: match-score(i, j) < match-score(k, j) and D(k) is greater than one, (d1 and d4 with g7, in Fig. 7).

In this case, we would not select the pair (k, j) if there is a column t, in row k such that match-score(k, t) > match-score(k, j). In this case, we would select the pair (i, j) instead. And we handle this case as in Case 1.

The example in Fig. 7, d1 and d4 match g7, a many-to-one conflict. The pair (d1, g7) is selected instead of (d4, g7) since the match-score(d4, g5) has a higher score than the pair (d4, g7).

A similar treatment is done if G(j) is greater than two. This step is repeated until no more D(i) is equal to one.

Step 5: Resolving the one-to-many conflicts.

For each D(i) that is a two, let j and k be the two entities in G-list that match with the i-th entity in the D-list. If match-score(i, j) >= match-score(i, k), we put the pair (i, j) in the one-to-one match list and set D(i) and G(j) to -1, and decrease G(k) by one. Otherwise, we put the pair (i, k) in the one-to-one match list and set D(i) and G(k) to -1, and decrease G(j) by one. A similar treatment is done if G(j) is greater than two. This step is repeated until no more D(i) is two or greater.

4.5 Computing Partial Matches: One-to-Many and Many-to-One

At the end of the one-to-one entity matching (Sect. 4.4), the value of each D(i) indicates whether the i-th entity has a one-to-one match. In particular, if a D(i) is a -1, it indicates that the i-th entity in D-file has a one-to-one match, otherwise, it indicates that it does not have a one-to-one match. (The same idea for the entities in G-file.) One could, for example, consider each D(i) >= 0 a false alarm and each G(j) >= 0 a miss detection.

However, it may be the case that a detected entity which does not have a one-to-one match may in fact match with a group of two or more groundtruth entities. For example, a detection algorithm may have located a text bounding box on the input image that includes several text lines, while those text lines are given one text bounding box each in the groundtruth file.

Therefore, to give partial credits to the detection algorithms for finding one-to-many partial matches, we do as follows.

For each D(i) >= 0 in the D-profile, we collect a list of all entities j in G-profile, such that G(j) is also >= 0 (also did not have a match by any D entity) and the score in the match-score(i, j) is greater than the *lower-threshold* (so that we would not include any noise). Currently, the *lower-threshold* is set to .05. However, we allow users to set their own threshold. If the sum of the scores for the entities in the collected list is greater than the *upper-threshold*, we consider the i-th D entity having a one-to-many partial match to those j entities in the collected list. And we set D(i) and all those G(j) to -1.

A similar protocol is applied to find the many-to-one matches (many detected entities matching with one groundtruth entity).

4.6 False-alarms and Miss-Detections

Finally, the false-alarms are those i entities having their D[i] values >= 0, and the miss-detections are those j entities having their G[j] values >= 0.

5 Matching Criteria

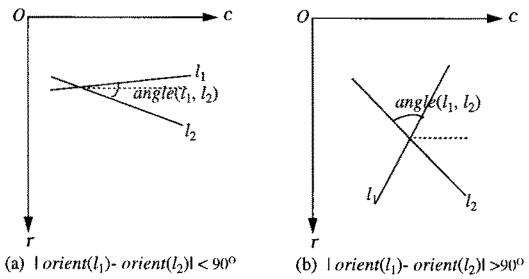
5.1 Line-Line Matching Protocol and Criteria

This protocol is for both solid lines and dashed lines. Let d be a line entity in the *D-entity-list* and g be a line entity in the *G-entity-list*. Let *match-score(i, j)* be the corresponding entry of d and g. To mark the entry (i, j), we compute the followings:

Step 1: If d and g have the same endpoints (a perfect match), we set match-score(i, j) to 1 and skip the following steps.

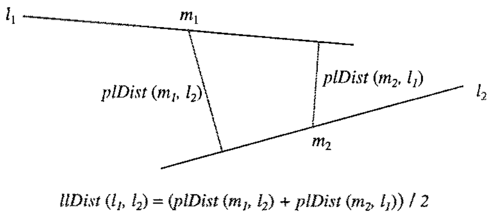
Step 2: We compute the angle between d and g as the included angle between these two line segments (see Fig. 8). If the *angle* between d and g is less than 5 degrees, $angle(d, g) \leq 5$, we continue to the next step, otherwise the *match-score(i, j)* is set to zero.

Step 3: We compute the *distance*, $llDist(d, g)$, between d and g. The $llDist(d, g)$ is computed as the average of the orthogonal distance from the midpoint of d to g and the orthogonal distance from the midpoint of g to d (see Fig. 9). If $llDist(d, g) \leq \theta_{ll}$, we continue to the next step, otherwise the *entity-match-table* is set to zero.



O
c
r
l_1
l_2
angle(l_1, l_2)
(a) | orient(l_1)- orient(l_2)| < 90°
(b) | orient(l_1)- orient(l_2)| >90°

Fig. 8. Angles of two line segments



l_1
m_1
l_2
m_2
plDist (m_1, l_2)
plDist (m_2, l_1)
llDist (l_1, l_2) = (plDist (m_1, l_2) + plDist (m_2, l_1)) / 2

Fig. 9. Line-line distance of two line segments

Step 4: Next, we compute the *relative overlap* function, $overlap(d, g, \alpha_g)$, of d and g with respect to the orientation of g. If $overlap(d, g, \alpha_g)$ is at least 20% of both d and g. we set match-score(i, j) to zero. Otherwise, we compute the relative overlap.
The relative overlap of two line segments l_1 and l_2 is defined as the ratio between the overlap function $overlap(l_1, l_2, \alpha)$ and the length of the longer segment: $RelativeOverlap(l_1, l_2, \alpha) = \frac{overlap(l_1, l_2, \alpha)}{\max(length(l_1), length(l_2))}$

Step 5: The match-score(i, j) is computed as:
$RelativeOverlap(l_1, l_2, \alpha) - \frac{angle(d,g)}{180} - \frac{llDist(d,g)}{\theta_{ll}}$.

For a precise definition of these functions see the Appendix A. The threshold values used here were determined heuristically based on the dimension of the entities. However, we allow these threshold values to be set by the user.

5.2 Arc-Arc Matching Protocol and Criteria

This protocol is for both solid arcs and dashed arcs.

Let A_1 be an arc entity in the *D-entity-list* and and A_2 be an arc entity in the *G-entity-list*. Let C_1 and C_2 be the centers of A_1 and A_2, and let R_1 and R_2 be the two radii (see Fig. 10).

Let match-score(i, j) be the entry that stores the matching result for this pair. The protocol for computing match-score(i, j) is as follows:

Step 1: If the two arcs, A_1 and A_2, are identical (with the same centers, same radii, same beginning and end angles), we set match-score(i, j) to 1 and skip the following steps.

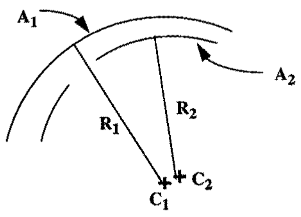


Fig. 10. An example of arc-arc entity pair

Step 2: We compute the point-to-point distance, $ppDist(C_1, C_2)$, between the two centers C_1 and C_2. If this distance is greater than θ_{CC}, we set match-score(i, j) to zero, and skip the following steps.

Step 3: We compute the absolute difference between the two radii, $RadiusDist(R_1, R_2) = |R_1 - R_2|$. If this distance is greater than θ_{RR}, we set match-score(i, j) to zero, and skip the following steps.

Step 4: We compute the ratio of these two radii, $RadiusRatio(R_1, R_2) = \frac{min(R_1,R_2)}{max(R1,R_2)}$. If this ratio is smaller than 85 percent, we set match-score(i, j) to zero, and skip the following steps.

Step 5: We project A_1 onto A_2, take the portion of A_2 that is within this projection, and call it A_3. The projection is defined as follows. We construct two lines, l_1 and l_2, from the center of A_1 to the two endpoints of A_1 to infinity. The wedge between these two lines (also within the beginning and the ending angles of A_1) is the projection of A_1. (See Fig. 11).

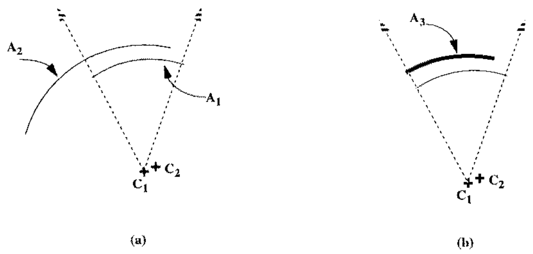


Fig. 11. (a) The projection of A_1 onto A_2, (b) A_3, the portion of A2 within the projection

Similarly, we project A_2 onto A_1 and take the portion of A_1 that is within this projection, call it A_4 (see part (d) in Fig. 12).

Step 6: If either A_3 or A_4 exists, we set the match-score(i, j) to zero and skip the rest. In the case that Both A_3 and A_4 exist, we define a line segment L_1 from the two ends of A_3 and another line segment L_2 from the two ends of A_4.

Step 7: Taking L_1 and L_2, we apply the line-line matching criteria to this pair.

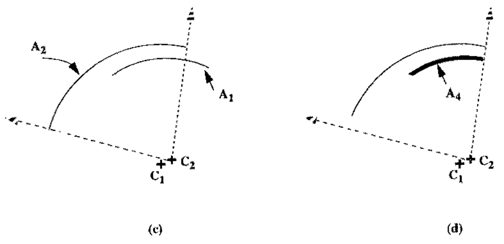


Fig. 12. (c) The projection of A_2 onto A_1, (d) A_4, the portion of A_1 within the projection

The match-score(i, j) for this pair is the result of the line-line matching result times F, an adjustment. Without lost of generality, let L_1 be shorter than L_2 and the distance between the two ends of A_2 be greater than that of A_1. Then F is computed as the ratio of the length of L_1 and the distance between the two ends of A_2.

5.3 Arc-Line Matching Protocol and Criteria

This protocol is for solid-arc with solid-line pairs and dashed-arc with dashed-line pairs.

Let A_1 be an arc entity in the *D-entity-list*, and let C_1 be the center and R_1 be the radius of A_1. Let L_1 be a line entity in the *G-entity-list* (see Fig. 13).

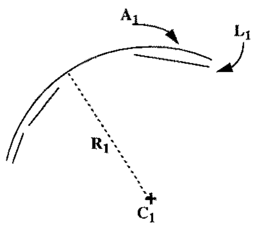


Fig. 13. An arc-line entity pair

Let match-score(i, j) be the entry that stores the matching result for this pair. The protocol for computing match-score(i, j) is as follows:

Step 1: We construct two lines, l_1 and l_2, from the center of A_1, to the two endpoints of L_2. We also construct a new line segment, R_2, from the center of A_1 to the midpoint of L_2. In the sense, we are constructing an artificial arc, A_2, taking C_1 as its center, R_2 as its radius, and the orientation of l_1 as its beginning angle, and the orientation of l_2 as its ending angles (see Fig. 14).

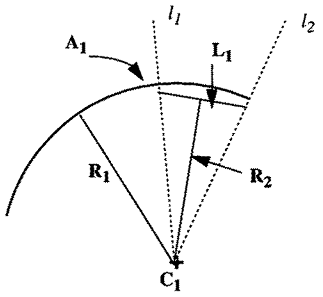


Fig. 14. Construction of the new triangle and the new line segment

Step 2: We compute the absolute difference between R_1 and R_2, $|R_1 - R_2|$. If this distance is greater than θ_{RR}, we set match-score(i, j) to zero, and skip the following steps.

Step 3: We project the artificial arc, A_2, onto A_1, take the portion of A_1 between l_1 and l_2, and call it A_3. If A_3 does not exist, we set match-score(i, j) to zero and skip the rest. In the case that A_3 exists, we construct a new line segment, L_1, from the new arc A_3 (see Fig. 15).

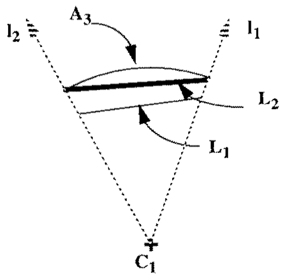


Fig. 15. The new line segment, L_1, which is constructed from A_3

Step 4: Taking L_1 and L_2 , we apply the line-line matching criteria to this pair. The match-score is set to the result of the line-line matching result times F, an adjustment. Without lost of generality, let L_1 be shorter than L_2 and the distance between the two ends of A_1 be greater than that of A_2. Then F is computed as the ratio of the length of L_1 and the distance between the two ends of A_1. If either L_1 or L_2 exist, we set the match-score(i, j) to zero.

5.4 Arc-Circle Matching Protocol and Criteria

This protocol is for solid-circle with solid-arc pairs and dashed-circle with dashed-arc pairs.

Let A_1 be an arc entity in the *D-entity-list* and and Cir be a circle entity in the *G-entity-list.* Let C_1 and C_2 be the centers of A_1 and Cir_2, and let R_1 and R_2 be the two radii.

Let match-score(i, j) be the entry that stores the matching result for this pair of entities. The protocol for computing match-score(i, j) is as follows:

Step 1: We project A_1 onto Cir_2, take the portion of the Cir_2 that is within this projection, and call it A_2. In the sense, we are constructing an artificial arc, A_2, taking C_2 as its center, R_2 as its radius, and the orientations of the two sides of the projection as its beginning and ending angles.

Step 2: We construct a line segment, L_1, from the two ends of A_1 and another line segment, L_2, from the two ends of A_2. Taking L_1 and L_2, we apply the line-line matching criteria to this pair. The match-score is set to the result of the line-line matching result times F, an adjustment. F is computed as the inside angle of A_1 divided by 360.

5.5 Circle-Circle Matching Protocol and Criteria

This protocol is for both solid and dashed circles.

Let Cir_1 be a circle entity in the *D-entity-list* and and Cir_2 be a circle entity in the *G-entity-list*. Let C_1 and C_2 be the two centers and let R_1 and R_2 be the two radii (see Fig. 16).

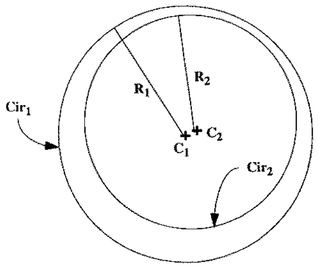


Fig. 16. A circle-circle entity pair

Let match-score(i, j) be the entry that stores the matching result for this pair of entities. The protocol for computing match-score(i, j) is as follows:

Step 1: If the two circles, Cir_1 and Cir_2, are identical (with the same centers and the same radii), we set match-score(i, j) to 1 and skip the following steps.

Step 2: We compute the point-to-point distance, $ppDist(C_1, C_2)$, between the two centers C_1 and C_2. If this distance is greater than θ_{CC}, we mark the entry as a non-match and skip the following steps.

Step 3: We compute the absolute difference between the two radii, $RadiusDist(R_1, R_2) = |R_1 - R_2|$. If this distance is greater than θ_{RR}, we mark the entry as a non-match and skip the following steps.

Step 4: If both $ppDist(C_1, C_2)$ and $RadiusDist(R_1, R_2)$ are equal to zero (two identical circles), we set match-score(i, j) to 1 and skip the following steps.

Step 5: We compute the ratio of these two radii, $RadiusRatio(R_1, R_2) = \frac{min(R_1,R_2)}{max(R1,R_2)}$. If this ratio is smaller than the preset threshold, we mark the entry as a non-match, otherwise, the entry is marked as a match.

Step 6: The matching score for this pair is

$$RadiusRatio(R_1, R_2) - \frac{ppDist(C_1, C_2)}{min(R1, R_2)} - \frac{RadiusDist(R_1, R_2)}{min(R1, R_2)}$$

5.6 Text-text Matching Protocol and Criteria

A text area is represented by a rectangular box and its orientation. The rectangular box is described by the x- and y- coordinates of any two opposite corners of the rectangle, and the orientation of the longest side of the rectangle.

Let t_1 be a text entity in the *D-entity-list* and let t_2 be a text entity in the *G-entity-list*. Let P_1 and P_2 be the two opposite corners of t_1 and let Q_1 and Q_2 be the two opposite corners of t_2.

Let match-score(i, j) be the corresponding entry of t_1 and t_2. The computational protocol for the match-score(i, j) is as follows:

Step 1: If t_1 and t_2 are identical (a perfect match), we set match-score(i, j) to 1 and skip the following steps.

Step 2: Let P_{mid} be the midpoint of the diagonal line of t_1 and let Q_{mid} be the midpoint of the diagonal line of t_2. Without lost of generality, let $ppDist(P_1, P_2) > ppDist(Q_1, Q_2)$. That is, t_1 has longer diagonal than that of t_2. We construct a circle centered at P_{mid} with radius equal to the diagonal of $t1$. If both the corner points of t_2, Q_1 and Q_2, are outside of this circle (this means that there is no overlap between t_1 and t_2), we set match-score(i, j) to zero, and skip the following steps. (The step is designed to limit the search space.)

Step 3: Next, we compute the other two opposite corners of t_1 and t_2, so that each of the text boxes is now represented as a rectangle. Let P and Q be the two rectangles.

Step 4: We compute the intersection of P and Q, call it, I. If I is empty, we set match-score(i, j) to zero. Otherwise, we compute the area of P, Q, and I. And we set $match - score(i, j) = \frac{area(I)}{max(area(P), area(Q))}$.

6 Appendix A: Line-Line Matching Criteria Functions

Image Coordinate System

An image is given by columns and rows of pixels. In a bi-level binary image, a foreground pixel has the value 1 and a background pixel has the value 0. We use the *Column-Row* coordinate system, (c-coordinate, r-coordinate), to represent

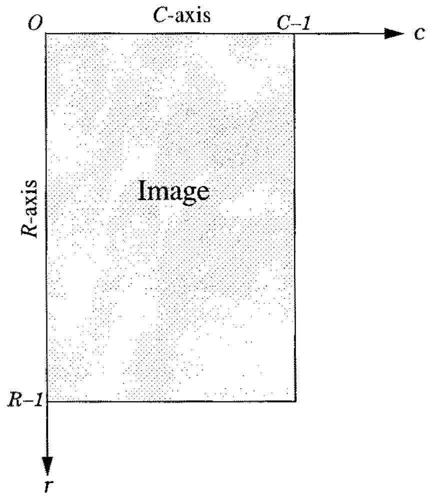


Fig. 17. The column-row coordinate system for an image. The origin (0,0) is at the top-left corner pixel of the image. The image has R rows and C columns.

a pixel's position within an image. The origin of this system, (0,0), is at the top-left corner pixel of the image (see Fig. 17).

Let d be a line entity in the *D-entity-list* and g be a line entity in the *G-entity-list*. The following functions are needed for the line-line matching criteria.

α-Projection of a Line Segment

The α-projection of a line $l = (c_1, r_1, c_2, r_2)$ is the projection of l onto the given orientation $\alpha \in (-90°, 90°]$. The α-projection of l, $proj(l, \alpha)$, is also a line segment, its two endpoints (c'_1, r'_1) and (c'_2, r'_2) are the projections of (c_1, r_1) and (c_2, r_2) onto the orientation α, respectively.

$$\begin{aligned}
c'_1 &= \cos\alpha(c_1 \cos\alpha + r_1 \sin\alpha) \\
r'_1 &= \sin\alpha(c_1 \cos\alpha + r_1 \sin\alpha) \\
c'_2 &= \cos\alpha(c_2 \cos\alpha + r_2 \sin\alpha) \\
r'_2 &= \sin\alpha(c_2 \cos\alpha + r_2 \sin\alpha)
\end{aligned}$$

If $|\alpha - orient(l)| \le 90°$, $proj(l, \alpha)$ is given by (c'_1, r'_1, c'_2, r'_2); otherwise, it is given by (c'_2, r'_2, c'_1, r'_1).

$$proj(l, \alpha) = \begin{cases} (c'_1, r'_1, c'_2, r'_2) \text{ if } |\alpha - orient(l)| \le 90° \\ (c'_2, r'_2, c'_1, r'_1) \text{ otherwise} \end{cases}$$

Overlap of Two Line Segments – A Relationship Function

The α-overlap of two line segment l_1 and l_2, $overlap(l_1, l_2, \alpha)$, is a relationship function of l_1 and l_2 with respect to a given orientation α.

Suppose T_a and T_d are two given thresholds, which are determined by application and user. T_a is a threshold for the angle of two line segments, T_d is a threshold for the line-line distance. If $angle(l_1, l_2)$ is not greater than T_a, and $llDist(l_1, l_2)$ is not greater than T_d, we say that l_1 and l_2 are sufficiently close.

The α-overlap of l_1 and l_2 is defined as the length of the common part of their α-projections if l_1 and l_2 are sufficiently close, and is defined as 0 otherwise. The function $overlap(l_1, l_2, \alpha)$ is

$$overlap(l_1, l_2, \alpha) = \begin{cases} length(proj(l_1, \alpha) \cap proj(l_2, \alpha)) & \\ \qquad \text{if } angle(l_1, l_2) \leq T_a, \text{ and } llDist(l_1, l_2) \leq T_d \\ 0 \qquad \text{otherwise} \end{cases}$$

Relative Overlap

The relative overlap of two line segments l_1 and l_2 is defined as the ratio between the overlap function $overlap(l_1, l_2, \alpha)$ and the length of the longer segment: $\frac{overlap(l_1, l_2, \alpha)}{\max(length(l_1), length(l_2))}$.

References

1. B. Kong, I. Phillips, R. Haralick, A. Prasad, and R. Kasturi. A benchmark: Performance evaluation of dashed line detection algorithms. In R. Kasturi and K. Tombre, editors, *Graphics Recognition: Methods and Applications, First International Workshop, University Park, PA, USA, August 1995, Selected Papers*, volume 1072 of *Lecture Notes in Computer Science*, pages 270–285. Springer, Berlin, 1996.
2. ECVNet. Benchmarking and Performance Evaluation web site. http://pandora.imag.fr/ECVNet/benchmarking.html.
3. O. Hori and S. Doermann. Quantitative measurement of the performance of raster-to-vector conversion algorithms. In R. Kasturi and K. Tombre, editors, *Graphics Recognition: Methods and Applications, First International Workshop, University Park, PA, USA, August 1995, Selected Papers*, volume 1072 of *Lecture Notes in Computer Science*, pages 57–68. Springer, Berlin, 1996.
4. L. Wenyin and D. Dori. A protocol for performance evaluation of line detection algorithms. *Machine Vision and Applications*, 9(5/6):240–250, 1997. Special Issue on Performance Characterisitics of Vision Algorithms.
5. L. Wenyin and D. Dori. A protocol for performance evaluation of algorithms for text segmentation from graphics-rich documents. In *Proceedings of Second IAPR Workshop on Graphics Recognition*, pages 317–324, Nancy, France, August 1997.
6. R. Haralick. Performance characterization in image analysis: Thinning, a case in point. *Pattern Recognition Letters*, 13:5–12, 1992.