A Performance Evaluation Protocol for Engineering-drawing Recognition Systems

Ihsin T. Phillips, Jisheng Liang†, and Robert Haralick†

Department of Computer Science/Software Engineering, Seattle University, Seattle, Washington 98122
†Department of Electrical Engineering, University of Washington, Seattle, Washington 98195
yun@seattleu.edu

Abstract

This paper defines a computational protocol to evaluate the performance of recognizers of solid line entities, circle entities, arc entities, dashed line entities, dashed circle entities, dashed arc entities, and text entities in engineering drawings. The protocol handles the one-to-many and many-to-one matching problems so that detected or groundtruth entities are not multiply counted.


1 Introduction

Systems which convert existing paper-based engineering diagrams into electronic format are in demand and a few have been developed. However, the performance of these 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 IAPR 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-lines test images and the corresponding groundtruth.

In this paper, we extend that protocol to evaluate the performance of engineering diagram recognition systems on images that contain engineering drawings of straight lines (solid or dashed), circles (solid or dashed), partial arcs of circles (solid or dashed), as well as bounding boxes of text blocks within the images.
Figure 1: 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 1 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 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. 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 mis-detections. A contingency table is produced. The contents of these tables are computed facts which when weighted by application specific weights can be summed to produce an overall score relevant to the application.

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”, “drawing-only”, or “all” option for their systems performance evaluations. The “text-only” evaluation option is designed for text block recognition systems that detect only the text blocks on the input image. The “drawing-only” evaluation option is designed for the recognition systems that detected only the drawing entities. The “all” evaluation option is designed for the systems that can detect both drawings and text blocks.

This paper is organized as follows: In section 2 we include the parameter definition for each of the seven entities (section 2.1) and the valid entity combinations (section 2.2). The protocols for the performance evaluation and the entity match computations are given in section 3. In section 4, we present the matching criteria for each pair of valid combinations. The precise definition of the line-to-line matching functions are given in Appendix A.
2 Entity Definition

2.1 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 (using the image coordinate system given in Appendix A) of the two end points (no special ordering for the two points), the thickness, and the orientation (in degree, clock-wise with respect to x-axis) of the line.

- 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 length of 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 length of the radius, the beginning and the ending angles (clock-wise, in degrees) of the arc, and the thickness 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 points of the rectangle, and the orientation of the longest side of the rectangle.

Although the evaluator is limited to the above seven entity types, nevertheless, it is useful, since all straightforward digital logic diagrams use only a combination of these geometric elements. Upgrading the evaluator is straightforward. To add another entity type, one needs to provide the evaluator the entity parameter information and the performance evaluation criteria.

2.2 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
3 Performance 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 mis-detections 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 section 3.1).

Step 4: Compute the one-to-one matches, resolving the problems of the one-to-many matches and the many-to-one matches, and compute the mis-detections and the false-alarms. (See section 3.2).

3.1 Matching Score Table Computation

The matching score table is computed as follows. We compute the matching scores (ranged 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 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 2 illustrates such a table. Blanks are read as zeros.

<table>
<thead>
<tr>
<th></th>
<th>g1</th>
<th>g2</th>
<th>g3</th>
<th>g4</th>
<th>g5</th>
<th>g6</th>
<th>g7</th>
<th>g8</th>
<th>g9</th>
<th>g10</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>d3</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>d5</td>
<td>86</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d6</td>
<td></td>
<td></td>
<td></td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d8</td>
<td>95</td>
<td>95</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: illustrates an example of a match-score table
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.

3.2 Computing Entity Matches

The protocol for computing the entity matches is as follows.

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 threshold, otherwise, it is set to zero. Currently, the threshold is set to .85. However, we allow the user to set its own threshold. Figure 3 illustrates the match-count table (on the right) which is computed from the match-score table on the left.

$$
\begin{array}{cccccccccc}
  & g1 & g2 & g3 & g4 & g5 & g6 & g7 & g8 & g9 & g10 \\
 d1 & . & . & . & . & . & . & . & . & . & . \\
d3 & . & . & . & . & . & . & . & . & . & . \\
d5 & . & . & . & . & . & . & . & . & . & . \\
d6 & . & . & . & . & . & . & . & . & . & . \\
d7 & . & . & . & . & . & . & . & . & . & . \\
d8 & . & . & . & . & . & . & . & . & . & . \\
\end{array}
$$

The interpretation of these two profiles can be as follows:

Figure 3: illustrates 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 4 illustrates the D-profile and the G-profile computed from the match-count table.

$$
\begin{array}{cccccccccc}
  & g1 & g2 & g3 & g4 & g5 & g6 & g7 & g8 & g9 & g10 \\
 d1 & . & . & . & . & . & . & . & . & . & . \\
d3 & . & . & . & . & . & . & . & . & . & . \\
d5 & . & . & . & . & . & . & . & . & . & . \\
d6 & . & . & . & . & . & . & . & . & . & . \\
d7 & . & . & . & . & . & . & . & . & . & . \\
d8 & . & . & . & . & . & . & . & . & . & . \\
\end{array}
$$

The interpretation of these two profiles can be as follows:

Figure 4: illustrates an example of the two projection profiles.
• 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.
(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 match 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-alarm: 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.

• Mis-detection: 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 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 Figure 5). 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 5 illustrates the result of one-to-one computation. There are two such pairs.

![Match-Score Table](image)

![Match-count Table](image)

Figure 5: illustrates results of one-to-one matches (circled).

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

For each D(i) that is equal to one (but did not find a one-to-one in Step 3), we locate the pair (i, j) such that match-count(i, j) is one, but G(j) is greater than one (d1 with g7, d6 and d8 with g2, in Figure 5).

If G(j) is a two (meaning that there are two D entities, say i-th and k-th, matching with the j-th entity in G-list),

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) is by one.
For example, if \( i = 6 \), we select the pair \((d6, g2)\) over \((d8, g2)\).

Case 2: We select the pair \((k, j)\) if \(\text{match-score}(k, j) > \text{match-score}(i, j)\) 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 \(D(i)\) is by one.

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

Case 3: \(\text{match-score}(i, j) < \text{match-score}(k, j)\) and \(D(k)\) is greater than one, \((d1\) and \(d4\) with \(g7\), in Figure 5).

In this case, we would not select the pair \((i, k)\) if there is an entry, say, \(t\), in row \(k\) such that \(\text{match-score}(i, t) > \text{match-score}(i, k)\). In this case, we would select the pair \((i, j)\) instead. And we handle this case as in Case 1.

The example in Figure 5, \(d1\) and \(d4\) match \(g7\), a many-to-one conflict. The pair \((1, 7)\) is selected instead of \((4, 7)\) since the \(\text{match-score}(4, 5)\) has a higher score than the pair \((4, 7)\).

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 \(\text{match-score}(i, j) \geq \text{match-score}(i, k)\), 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.

Step 6: Compute the false-alarm list. We put each entry in the \(D\)-profile having a zero or greater value to the false-alarm list since we could not find a match for it (otherwise, the entry would be -1.)

Step 7: Compute the mis-detection list. We put each entry in the \(G\)-profile having a zero to the mis-detection list since we could not find a match for it (otherwise, the entry would be -1.)

4 Matching Criteria

4.1 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 entity-match-score-table \((i, j)\) be the corresponding entry of \(d\) and \(g\). To mark the entry \((i, j)\), we compute the followings:

Step 1: We compute the angle between \(d\) and \(g\) as the included angle between these two line segments (see Figure 6). If the angle between \(d\) and \(g\) is less than 5 degrees, \(\text{angle}(d, g) \leq 5\), we continue to the next step, otherwise the entity-match-score-table \((i, j)\) is marked as a non-match.
Step 2: 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 Figure 7). If $llDist(d, g) \leq \theta_{ll}$, we continue to the next step, otherwise the entity-match-table is marked as non-match.

Step 3: Next, we compute the relative overlap function, $\text{overlap}(d, g, \alpha_g)$, of $d$ and $g$ with respect to the orientation of $g$. If $\text{overlap}(d, g, \alpha_g)$ is at least 20 percent of both $d$ and $g$, we marked the entry for the pair in the entity-match-table as a match, and a non-match otherwise, marked as non-match.

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

$$\text{relative} - \text{overlap}(l_1, l_2, \alpha) = \frac{\text{overlap}(l_1, l_2, \alpha)}{\max(\text{length}(l_1), \text{length}(l_2))}$$

Step 4: The match score for the entity-match-score-table $(i, j)$ is computed as:

$$\text{Entity - match - score - table}(i, j) = \text{relative} - \text{overlap}(l_1, l_2, \alpha) + \frac{\text{angle}(d, g)}{180} - \frac{llDist(d, g)}{\max(\text{length}(l_1), \text{length}(l_2))}$$

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.

4.2 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 $A_2$ 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 Figure 8).

The protocol for computing the match for this pair is as follows:

Step 1: 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 $\theta_{CC}$, we mark the entry as a non-match and skip the following steps.

Step 2: We compute the absolute difference between the lengths of the two radii, $RadiusDist(R_1, R_2) = |R_1 - R_2|$. If this distance is greater than $\theta_{RR}$, we mark the entry as a non-match and skip the following steps.

Step 3: We compute the ratio of these two radii, $RadiusRatio(R_1, R_2) = \frac{\min(R_1, R_2)}{\max(R_1, R_2)}$. If this ratio is smaller than 80 percent, we mark the entry as a non-match and skip the following steps.

Step 4: We construct two lines, $l_1$ and $l_2$, from the two endpoints of $A_2$ and $C_1$, the center of $A_1$. If both $l_1$ and $l_2$ intersect $A_1$, then we take the portion of arc segment of $A_2$ between $l_1$ and $l_2$, call it $A_3$ (see part (b) in Figure 9).

Similarly, we construct two lines, $l_3$ and $l_4$, from the two endpoints of $A_1$ and $C_2$, the center of $A_2$. If both $l_3$ and $l_4$ intersect $A_2$, then we take the portion of arc segment of $A_1$ between $l_3$ and $l_4$, call it $A_4$ (see Figure 10). Note that, in Figure 10, $A_4$ does not exist since either $l_3$ or $l_4$ intersects $A_2$.

Step 5: In the case that $A_3$ exists, we construct two line segments, $L_1$ and $L_2$, from the two arcs, $A_1$ and $A_3$, by connecting the two endpoints of each arc (see Figure 11). Similarly, if $A_4$ exists, we construct another pair of line segments, $L_3$ and $L_4$, from the two arcs, $A_2$ and $A_4$ (if $A_4$ exists), by connecting the two endpoints of $A_2$ and $A_4$. 

Figure 8: illustrates an example of an arc-arc pair entities.

Figure 9: (a) illustrates the two new lines which are constructed from the two endpoints of $A_2$ and $C_1$. (b) illustrates the construction of the new arc $A_3$. $A_3$ is the portion of $A_1$ between the lines $l_1$ and $l_2$. 

Figure 10: (a) illustrates the construction of the new arc $A_4$. $A_4$ is the portion of $A_2$ between the lines $l_3$ and $l_4$.
Figure 10: illustrates the two new lines, \( l_3 \) and \( l_4 \) are constructed from the two endpoints of \( A_1 \) and \( C_2 \). However, since neither \( l_3 \) or \( l_4 \) intersects \( A_2 \), the new arc segment, \( A_4 \) can not be constructed.

Figure 11: illustrates the two new line segments, \( L_1 \) and \( L_2 \) are constructed from the two arcs of \( A_1 \) and \( A_3 \).

Step 6: Taking \( L_1 \) and \( L_2 \), if they exist, we apply the line-line matching criteria to this pair. Then taking \( L_3 \) and \( L_4 \), if they exist, we apply the line-line matching criteria to this pair. We check the results of the two line-line matchings. If any of the two line-line matchings results in a non-match, we mark the table entry corresponding to the two arcs, \( A_1 \) and \( A_2 \), as a non-match. Otherwise, we mark the table entry as a match.

4.3 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 \( L_2 \) be a line entity in the \( G\)-entity-list. (see Figure 12).

Figure 12: illustrates an example of an arc-line pair entities.

The protocol for computing the match for this pair is as follows:

Step 1: We construct two lines, \( l_1 \) and \( l_2 \), from the two endpoints of \( L_2 \) and the center of \( A_1, C_1 \). We also construct a new line segment, \( R_2 \), by connecting the midpoint of \( L_2 \) and \( C_1 \). The two endpoints of \( L_2 \) and \( C_1 \) form a triangle (see Figure 13).
Step 2: We compute the absolute difference between the lengths of $R_1$ and $R_2$, $|R_1 - R_2|$. If this distance is greater than $\theta_{RR}$, we mark the entry as a non-match and skip the following steps.

Step 3: We compute the angle difference between the two base angles, $a_1$ and $a_2$, of the new triangle. If this difference is too large, we mark the entry as a non-match and skip the following steps.

Step 4: If both $l_1$ and $l_2$ intersect $A_1$, then we take the portion of the arc segment of $A_1$ between $l_1$ and $l_2$, call it $A_3$. In the case that $A_3$ exists, we construct a new line segment, $L_1$, from the new arc $A_3$, by connecting the endpoints of $A_3$ (see Figure 14).

Step 6: Taking $L_1$ and $L_2$ if they exist, we apply the line-line matching criteria to this pair. If this line-line matching returns a non-match, we mark the table entry corresponding to the arc, $A_1$ and the line $L_2$ as a non-match. And mark as a match otherwise.

4.4 Circle-Arc Matching Protocol and Criteria

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

The matching procedure and matching criteria for matching a circle entity to an arc entity is identical to the matching procedure for the arc-to-arc protocol.
4.5 Circle-Circle Matching Protocol and Criteria

This protocol is for both solid and dashed circles.

Let \( C_1 \) be a circle entity in the D-entity-list and and \( C_2 \) 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 Figure 15).

![Figure 15: illustrates an example of an arc-arc pair entities.](image)

The protocol for computing the match for this pair is as follows:

Step 1: 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 \( \theta_{CC} \), we mark the entry as a non-match and skip the following steps.

Step 2: We compute the absolute difference between the lengths of the two radii, \( RadiusDist(R_1, R_2) = |R_1 - R_2| \). If this distance is greater than \( \theta_{RR} \), we mark the entry as a non-match and skip the following steps.

Step 3: We compute the ratio of these two radii, \( RadiusRatio(R_1, R_2) = \frac{\min(R_1, R_2)}{\max(R_1, R_2)} \). If this ratio is smaller than 80 percent, we mark the entry as a non-match, otherwise, the entry is marked as a match.

Step 4: The matching score for this pair is

\[
RadiusRatio(R_1, R_2) = \frac{ppDist(C_1, C_2)}{\min(R_1, R_2)} - \frac{RadiusDist(R_1, R_2)}{\min(R_1, R_2)}
\]

4.6 Text-text matching

The matching score for a pair of text areas is simply computed as the ratio of the area of their intersection to the area of their union.

5 Appendix A: Line-Line Matching Criteria Functions

Image coordinate system

An image is given by columns and rows of pixels. In an 8-bit binary image, a foreground pixel has the value 255 and a background pixel has the value 0. We use the Column-Row coordinate system, (c-coordinate, r-coordinate), to represent 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 Figure 16).
Figure 16: illustrates the Column-Row coordinate system on an image. The origin (0,0) is at the top-left corner pixel of the image. The image has $R$ rows and $C$ columns.

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 is 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 $α \in (-90^\circ, 90^\circ]$. The $α$-projection of $l$, $proj(l, α)$, 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.

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

If $|α - orient(l)| \leq 90^\circ$, $proj(l, α)$ is given by $(c'_1, r'_1, c'_2, r'_2)$; otherwise, it is given by $(c'_2, r'_2, c'_1, r'_1)$.

$$
proj(l, α) = \begin{cases} 
    (c'_1, r'_1, c'_2, r'_2) & \text{if } |α - orient(l)| \leq 90^\circ \\
    (c'_2, r'_2, c'_1, r'_1) & \text{otherwise}
\end{cases}
$$

**Overlap of two line segments —— a relationship function**

The $α$-overlap of two line segment $l_1$ and $l_2$, $overlap(l_1, l_2, α)$, 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 sufficient 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 sufficient close, and is defined as 0 otherwise. The
function \textit{overlap}(l_1, l_2, \alpha) is

\[
\text{overlap}(l_1, l_2, \alpha) = \begin{cases} 
CRlen(proj(l_1, \alpha) \cap proj(l_2, \alpha)) & \text{if angle}(l_1, l_2) \leq T_a, \text{ and } ||Dist(l_1, l_2) \leq T_d} \\
0 & \text{otherwise}
\end{cases}
\]

\textbf{Relative overlap}

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

\[
\frac{\text{overlap}(l_1, l_2, \alpha)}{\max(\text{length}(l_1), \text{length}(l_2))}
\]

\textbf{References}