Abstract

Currently, there are large collections of clip-art vector drawings from which users can select the desired figures to insert in their documents. However, to locate a particular drawing among thousands is not easy. Although there are some solutions for drawing retrieval, almost all of them are designed to retrieve simple and not complex drawings as for instance clip-arts. In our prior work we proposed an approach to index and retrieve complex vector drawings by content, using topological and geometric information automatically extracted from figures. In this paper, we present a new algorithm to improve the geometric matching of two drawings, which takes into account the drawing as a whole and each of the shapes in it. We developed a web search engine for clip-art drawings, where we included this new technique. Experimental evaluation reveals that this new geometric matching algorithm conducts to better retrieval results than the prior matching solution.

Key words: Geometric Matching, Drawing Retrieval, Sketch-Based Retrieval

1 Introduction

Nowadays, there are a lot of vector drawings available for integration into documents, either on the Internet or on clip-art collections sold in optical media. This large number of drawings makes traditional searching mechanisms, based on browsing and navigation in directories or complex mazes of categories, inadequate. Furthermore, solutions using keywords or tagging are also impracticable since they have to be generated manually. A more adequate solution must take into account information automatically extracted from the content of drawings, instead of information manually generated by people. Although there are several solutions for Content-Based Image Retrieval (CBIR),
they cannot be applied to vector drawings, because these are described in a structural manner, requiring different approaches from those used for raster images. Currently there are a small set of solutions for vector drawings, but either they only deal with simple drawings (mainly contours) or they only support small sets of elements in the database.

In our prior work [1], we proposed an automatic visual classification scheme based on topology and geometry, to retrieve large sets of vector drawings. Our solution takes advantage of users’ visual memory and explores their ability to sketch as a query mechanism. We used a graph-based technique to describe the spatial arrangement of drawing components, coding topological relationships of inclusion and adjacency through the specification of links between nodes of the graph. We used a multidimensional indexing method that efficiently supports large sets of drawings, in combination with new schemes that allow us to hierarchically describe drawings and subparts of drawings by level of detail. This way we are able to perform searches using coarse approximations or parts of the desired drawing. Matching in this solution was performed in two steps: first we selected the drawings with similar topology to the query, then we compared the geometry of this subset of topologically similar drawings. Although, this matching sequence produced good results for technical drawings [2], which have a richer topology, the same does not apply to clip-art drawings that are more geometric than topological. Indeed, during user evaluation, we observed that users, while searching for clip-art drawings, typically draw a small number of shapes, and consequently do not specify topology, but only geometry. Moreover, our most recent experimental evaluation [3] revealed that this topology plus geometry combination produces poor results for retrieving clip-art drawings using sketches as queries. Additionally, we identify the need to improve the geometry comparison algorithm, since the topology filter “is not doing its job”. While with CAD drawings we can have a more flexible geometric matching algorithm, with clip-art drawings we must have a tighter geometric comparison mechanism to assure good results even without the use of topology.

In this paper we explain the possible reasons of the poor performance of our prior matching algorithm, and we propose a new geometric matching mechanism, bearing in mind the importance of the human visual perception. To that end we use the similarity of drawings as a whole and the similarity between each individual shape in drawings. We developed a prototype for the retrieval of clip-art drawings integrating both algorithms, the prior and the new (see Figure 1). This prototype was used to evaluate the performance of the two algorithms. Experimental results revealed that our new approach to compare the geometry of drawings outperforms the prior solution.

The rest of the paper is organized as follows: Section 2 provides an overview of related work in content-based retrieval of drawings. In Section 3 we present
an overview of our framework for sketch-based retrieval of drawings. Section 4 describes our previous geometric matching technique, allowing readers to understand the differences to the new matching algorithm, which we present in Section 5. In Section 6, we describe the experimental evaluation performed to compare both approaches. Finally, in Section 7 we conclude and enumerate some future work.

2 Related Work

In the past years there has been a great focus in querying Multimedia databases by content. However, most such work has focused on image databases disregarding the retrieval of vector drawings. Due to their structure vector drawings require different approaches from image-based methods, which resort to color and texture as main features to describe content. While some early work [4,5] attempted to index drawings through textual databases, such approaches do not take advantage of the rich visual association mechanisms and drafts-people use of sketches to recover information.

A current trend in multimedia information processing is toward Content-Based Retrieval. Rather than manually generate text-based descriptions, content-based retrieval works by matching the query against an automatically generated representation of the content of the element to retrieve. In the last decade, the majority of indexing and retrieval systems were developed for digital images. Vector drawings, which require different approaches, only in recent years received attention from researchers.

One of the first works dedicated to the retrieval of drawings was Gross’s Electronic Cocktail Napkin [6]. This system addressed a visual retrieval scheme
based on diagrams, to indexing databases of architectural drawings. Users draw sketches of buildings, which are compared with annotations (diagrams), stored in a database and manually produced by users. Even though this system works well for small sets of drawings, the lack of automatic indexation and classification makes it difficult to scale the approach to real collections of drawings.

Berchtold and Kriegel developed the S3 system [7] to support managing and retrieving industrial CAD parts. Drawings are described by their geometry (2D contour) and thematic attributes. The S3 system relies exclusively on matching contours, ignoring spatial relations and shape information, making this method unsuitable for retrieving complex multi-shape drawings.

Another approach for retrieving engineering CAD drawings was developed by Müller and Rigoll, based on stochastic models [8]. Drawing databases are searched using sketches or shapes which represent details in mechanical parts. Their approach aims to retrieve images containing certain specified details and locating these details in the retrieved images. Authors represent drawings and queries using a pseudo 2-D Hidden Markov Model augmented with filler states. Their method only supports simple queries, representing a single element. More complex queries including several elements with spatial relationships between them are not contemplated. Furthermore, the search mechanism is not appropriate for large collections of drawings, since they perform a sequential scan through the database comparing the query with all indexed drawings.

A system that supports the retrieval of complex 2D drawings was presented in 1999 by Park and Um [9]. Their approach is based on dominant shapes, where objects are described by recursively decomposing its shape into a dominant shape, auxiliary components and their spatial relationships (inclusion and adjacency). These and the decomposition of blocks are stored in a complex graph structure. Visual elements in drawings are classified using a set of geometric primitives (triangles, circles, etc.). However, this small set of base geometric primitives and the not-so-efficient matching algorithm, based on the breadth-first tree matching, make it hard to handle large databases of drawings.

In the work of Beretti and Del Bimbo [10], authors describe shapes from a drawing by decomposing them into tokens that correspond to protrusions of the curve. To compute the similarity between shapes, authors verify if the two shapes share tokens with similar curvature and orientation, within a given threshold. However, the efficiency of the similarity computation depends on the number of tokens in each shape and does not take into account the token order.

Leung and Chen proposed a sketch retrieval method [11] for general unstruc-
tured free-form hand-drawings stored in the form of multiple strokes. They use shape information from each stroke exploiting the geometric relationship between multiple strokes for matching. Later on, authors improved their system by also considering spatial relationships between strokes [12]. Authors use a graph based description, similar to ours, but describing only inclusion, while we also describe adjacency. Their technique has two drawbacks, complexity, since they use a restricted number of basic shapes (circle, line and polygon) and scalability.

Another approach for matching hand-drawn sketches is the line-based representation of sketches proposed by Namboodiri and Jain [13]. To skirt around the problem of identifying basic shapes from a sketch, drawings are represented as a set of straight lines. While the algorithm is simple to implement it presents scalability problems to more complicated drawings and larger datasets, since it entails sequential search and one-to-one comparisons. Moreover, the conversion into straight lines is very dependent of the way users draw sketches.

Mascio et al. [14] presented an approach for sketch-based retrieval of drawings using a CBIR engine. Although they retrieve vector drawings, authors first convert them into raster images and then apply a set of CBIR techniques. Although authors do not present any evaluation of their solution, we think that converting vector drawings to raster images is not the best approach, since it consumes time and causes the lost of information during the process.

Liang et al. [15] developed a solution for drawing retrieval based on our prior solution [1]. Authors included some differences, such as the use of eight topological relationships and relevance feedback. Additionally, they segment sketches using vertices, drawing speed and curvature. By using eight topological relationships, the description and comparison will be more restrictive, producing less results, and reducing recall.

Nabil et al. [16] presented a set of techniques for similarity retrieval based on the 2D Projection Interval Relationships representation (2D-PIR), which include methods for dealing with rotated and reflected images. It adapts three existing representation formalisms (Allen’s temporal intervals, 2D-strings and topological relationships) combining them in a novel way to produce a unified representation of pictures. Authors claim that their method offers more information about spatial relationships between objects in a picture than traditional methods. However, during the matching process the symbolic representation of the query gets compared to all the symbolic representations stored in the database, making this work difficult to scale up for large collections of images.

Pu and Ramani, developed an approach that analyzes drawings as a whole [17]. Authors proposed two methods to describe drawings. One uses the 2.5D
spherical harmonics to convert a 2D drawing into a 3D representation, which is independent to rotations. The other method, the 2D shape histogram, creates a signature with the shape distribution, by computing values for points in the surface of the shape. This method is independent of transformations, insensible to noise, simple and fast. After experimental evaluation, authors decided to combine both methods to get a better descriptor and to increase the system accuracy.

Recently Hou and Ramani [18] presented an approach for contour shape matching of engineering drawings, inspired by the divide and conquer paradigm. They divide the original shape into two levels of representation, a higher level with structure and a lower level with geometry. During matching, they first use the structure level and then the geometry level, to find similar shapes.

From the content-based retrieval systems described above we can observe two things: First, most published works rely mainly on the geometric description of drawings (mainly contours), discarding the spatial arrangement of drawing items. Second, the majority of the solutions deal with simple drawings using just their contour to describe the geometry, as illustrated in the samples presented in Figure 2. These are not suitable for retrieving clip-art drawings, since they are more complex and have a bigger number of visual elements, requiring other mechanisms to describe and compare their geometry. Moreover, most of this solutions rely on query-by-example approaches, which are not as challenging as query-by-sketch.

In this paper, we describe and evaluate a solution for sketch-based retrieval of clip-art drawings, putting the focus on a new approach for geometric matching between simple sketched queries and complex clip-art drawings. This approach tries to solve the problems of simplicity of drawings and lack of spatial information, identified in existent solutions.

3 Framework Overview

The new algorithm developed to compare the geometry between queries and drawings was integrated in our general framework for sketch-based retrieval
of drawings, developed previously [1]. This framework aims to retrieve vector drawings privileging the use of spatial relationships and geometric information. Indeed, our solution is more ambitious than existing solutions, in the sense that we do automatic simplification, classification and indexation of existing drawings, to make the retrieval process of complex drawings both more effective and accurate. These activities imply specifying a description mechanism for complex drawings and simple sketched queries. Additionally, fast and efficient algorithms to perform similarity matching between sketched queries and a large database of drawings are required.

To give context to the reader and to explain some of the topics needed to describe our new geometric matching approach, we shortly present an overview of the overall framework, describing its main components (see Figure 3).

### 3.1 Classification

In the context of vector drawings, features such as color and texture, used mainly in the domain of digital images, are not very expressive. Instead, features related to the shape of objects (geometry) and to their spatial arrangement (topology) are more descriptive of drawing contents. So, in our framework we focus on topology and geometry as main features. Although, clip-art drawings include color, data we collected from experiments with users highlighted two things: One, when users are searching for drawings (clip-arts
or technical drawings) they are not very worried about the position in the returned data where the results appear (precision), they only want that the drawings they are looking for appear in the results (recall). So, by using color in the description of both drawing contents and query specifications we risk increasing precision at the cost of reducing recall. Second, in a previous prototype we used color to describe clip-arts. Not only our experimental tests with users reveal poorer results, we also noticed that they almost never specify color while searching for drawings, even when such an option is made available to them.

Our classification process starts by applying a simplification step, to eliminate most useless elements. The majority of drawings contain many details, which are not necessary for a visual query and increase the cost of searching. We try to remove visual details (i.e. small polygons and lines) while retaining the perceptually dominant elements and shapes in a drawing. This way we reduce the number of entities to analyze in subsequent steps of the classification process, speeding up queries.

3.1.1 Topology

After simplification we identify visual elements, namely polygons and lines, and extract geometric and topological information from drawings. We use two relationships, **Inclusion** and **Adjacency**, which are a simplified subset of the topological relationships defined by Egenhofer [19]. While these relationships are weakly discriminating, they do not change with rotation and translation. Relationships thus extracted are compiled in a **Topology Graph**, where ”vertical” edges mean Inclusion and ”horizontal” connections mean Adjacency, as illustrated in Figure 4. Our topology graph has a well defined structure, being very similar to ”a rooted tree with side connections”. It has always a root node, representing the whole drawing. Sons from the root represent the dominant blocks (polygons) from the drawing, i.e. blocks that are not contained in any other block. The next level of the graph describes polygons contained by the blocks identified before. This process is applied recursively until we get the complete hierarchy of blocks.

![Fig. 4. Drawing and Topology graph.](image-url)
Since graph matching is a NP-complete problem, we are not directly using topology graphs for searching similar drawings. We use the corresponding graph spectra instead. For each topology graph to be indexed in a database we compute descriptors based on its spectrum [20]. To generate the graph spectrum we first create the adjacency matrix of the graph, second we calculate its eigenvalues and finally we sort the absolute values to obtain the topology descriptor (see Figure 5). The resulting descriptor is a multidimensional vector, whose size depends on graph (and corresponding drawing) complexity. Very complex drawings will yield descriptors with higher dimensions, while simple drawings will result in descriptors with lower size.

![Topology Graph Adjacency Matrix Eigenvalues Topology Descriptor](image)

Fig. 5. Block diagram for topology descriptor computation.

In this way, we reduce the problem of isomorphism between topology graphs to computing distances between descriptors. To support partial drawing matches, we also compute descriptors for sub-graphs of the main graph. Moreover, we use a new way to describe drawings hierarchically, by dividing them in different levels of detail [1] and then computing descriptors at each level. This combination of sub-graphs descriptors and levels of detail, provides a powerful way to describe and search both for drawings or sub-parts of drawings, as illustrated in Figure 6.

![Fig. 6. Finding drawings by providing a sub-part of it.](image)

While this solution produced good results in the past, we notice that in some cases results could be improved if we take into account the distance between the visual elements in a drawing. To that end, we recently devised a new mechanism to include proximity into our topology graph [3]. This way, we are able to differentiate between a drawing with two polygons which are close

![Fig. 7. Using the adjacency weight to differentiate between far and near objects.](image)
together and a drawing with two polygons that are far apart, as illustrated in Figure 7.

To code proximity in the topology graph, we associate weights to the adjacency links of the graph. While in our previous solution we only have an adjacency link when two primitives are connected, now we compute the (normalized) distance between two elements and use this value as the weight of the link. This change in the weights of the topology graph does not affect the stability and robustness of eigenvalues, as ascertained by Sarkar and Boyer [21].

3.1.2 Geometry

While topology convey global information about the drawing, the shape of an object represents local characteristics which can be used to narrow down the search.

To describe the geometry of entities from vector drawings, we use a general shape recognition library which is able to identify a set of geometric figures and gestural commands called CALI [22]. In our approach instead of using CALI to recognize a shape or a gestural command from polygons, we compute a set of geometric features and store them in a multidimensional vector (see Figure 8).

\[
\begin{bmatrix}
P_{ch} & A_{ch} & H_{ew} & A_{ew} & A_{e} & A_{ch} & A_{q} & A_{c} & P_{ch} & P_{eh} & P_{eh} & P_{er}
\end{bmatrix}
\]

Fig. 8. Geometric feature vector.

We start the calculation of these geometric features by computing the Convex Hull of the provided element (visual element from a drawing or a sketch).

Fig. 9. Special polygons of a geometric entity.
Then, we compute three special polygons from the convex hull: the Largest Area Triangle and the Largest Area Quadrilateral inscribed in the convex hull, and finally, the Smallest Area Enclosing Rectangle, as illustrated in Figure 9.

Finally, we compute the ratios between area and perimeter from each special polygon, to create the multidimensional feature vector presented in Figure 8. Table 1 lists these ratios and the similarity that they measure. We experimentally evaluated several ratios, as described in [23], before we reach this set.

Table 1
Features used to describe the geometry of objects.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{P_{ch}}{A_{ch}}$</td>
<td>This feature measures the complexity of shapes, i.e. it measures the length of strokes within a limited area (convex hull).</td>
</tr>
<tr>
<td>$\frac{A_{ch}}{P_{ch}}$</td>
<td>This feature measures the similarity to a circular shape.</td>
</tr>
<tr>
<td>$\frac{H_{wr}}{W_{wr}}$</td>
<td>The aspect ratio measures the thinness of shapes.</td>
</tr>
<tr>
<td>$\frac{A_{ch}}{A_{er}}$</td>
<td>This set of features measures the similarity to a rectangle.</td>
</tr>
<tr>
<td>$\frac{P_{ch}}{P_{er}}$</td>
<td></td>
</tr>
<tr>
<td>$\frac{A_{lq}}{A_{lt}}$</td>
<td>This set of features measures the similarity to a triangle.</td>
</tr>
<tr>
<td>$\frac{P_{lq}}{P_{lt}}$</td>
<td></td>
</tr>
</tbody>
</table>

These features allow the description of shapes independently of their size, rotation, translation or line type, and can either be used to classify drawings or hand-sketched queries. Using geometric features instead of polygon classification, allows us to index and store potentially unlimited families of shapes. We compared the discriminative power of our geometry vector with five other approaches, namely Zernike Moments, Fourier descriptors, grid-based, Delaunay triangulation and Touch-point-vertex-angle-sequence, as described in [24]. Results show that our approach outperforms all other algorithms, including for the most part, the Zernike moments.

In summary, at the end of the classification step we have, for each drawing, a set of topological descriptors that describe the topology of the complete drawing and of its subparts, and a set of geometric descriptors (one for each visual element in the drawing) that characterize the drawing geometry.
3.2 Query and Indexing

Our framework includes support for calligraphic interfaces, allowing the specification of queries using sketches. This type of queries raises new problems that are not present in query-by-example systems, such as, the comparison of simple queries with complex drawings. Our query component performs the same steps as the classification process, namely simplification, topological and geometric feature extraction, topology graph creation and descriptor computation. In an elegant fashion two types of information (vector drawings and sketches) are processed by the same pipeline.

To improve the searching performance while using large databases of drawings, we use a multidimensional indexing structure in our framework. This indexing structure, the NB-Tree [25], is a simple, yet efficient indexing structure, which uses dimension reduction. It maps multidimensional points to a 1D line by computing their Euclidean Norm. To overcome the lost of information during dimension reduction, we insert the original multidimensional index vector into the corresponding B+-Tree node and use the norm as the key to perform searches. This way, we do not need to reconstruct the original vector, after we obtain the search results.

3.3 Matching

Computing the similarity between a hand-sketched query and all drawings in a database can entail prohibitive costs especially when we consider large sets of drawings. Thus, in our previous solution, we divide our matching process in a two-step procedure, to speed up searching. The first step relies on the global feature extracted from drawings, topology. It searches for topologically similar drawings, performing a KNN search in the indexing structure with topological information. This works as a first filter to avoid unnecessary geometric matches between false candidates. In the second step we compare the local geometric information from the query to that of drawings that passed the topology step. At the end of the matching process we get a measure of similarity, that combines topology and geometry, between the sketched query and drawings retrieved from the database.

In this matching algorithm we assumed that topology plays an important role on the description and filtering of drawings. This is true for technical drawings, as we observed in previous tests with users [26], where they explore the spatial arrangement of query elements to convey more information. However, for clip-art drawings we found out that users specify more geometry than spatial arrangements, invalidating the first filter by topology. Due to this, we had the
need to improve our previous geometric matching algorithm (Section 4), by developing a new approach, which we describe in Section 5.

4 Previous Matching Algorithm

Our prior algorithm for geometry matching assumed that most of the non relevant drawings were discard by the topology filter. So, this reduced set of drawings were then compared to the sketched query, yielding a measure of geometric similarity for each drawing. The similarity measure was obtained through the computation of distances between geometric query descriptors and geometric data stored in an indexing structure. In our approach each drawing has its own indexing structure with the geometric information, to simplify the matching process. Thus, during polygon comparison we only compute similarity distances between relevant descriptors.

The matching procedure starts by computing a geometric descriptor for each entity specified in the query, using the CALI library. Then, each of these descriptors are used to perform a KNN search (being K the number of polygons in the sketched query) to each geometric indexing structure, one for each topologically similar drawing. Returned results have a distance associated that convey similarity. Smaller distances mean more similar while larger distances mean less similar. Finally, we iteratively select the pair of polygons (one from the query and other from the drawing) that has the smallest distance. After each selection the pair of polygons is eliminated from the list of results.

We will now explain in detail how we compute the geometric similarity measure resorting to an example. Consider that we have a query with four polygons and a drawing in the database with five polygons, belonging to a drawing. After performing four 4-NN queries to the correspondent indexing structure (one for each visual entity in the query), we build a matrix representing distances from each polygon in the query (qP1–qP4) to all polygons in the drawing (dP1–dP5), as depicted in Figure 10 (top-left).

To compute the similarity value, we first find the row of the matrix with the smallest value (0.1). Then, we store this value and delete the corresponding row (qP1) and column (dP5). From the remaining matrix we perform the same steps, selecting the new smallest value (0.12) and deleting row qP2 and column dP2. We continue this iterative process until we have selected values for all polygons of the query. Finally, we sum all selected values and divide by the number of polygons in the query. When the drawing has less polygons than the query, we compute values for the query polygons most similar to those in the drawing and divide the sum by the number of polygons in the drawing. This process of computing the geometric similarity value is then applied to
the remaining drawings that passed through the topology filtering, yielding a list of drawings with a geometric measure of similarity to the sketched query.

This method does not return the "optimum" result, since it does not minimize the similarity value, but on the other hand it is simple and very fast to compute. To obtain the optimum value, we could use the Hungarian method [27], which works in polynomial time.

One of the drawbacks of our first solution was the fact that it does not take into account the number of elements in the query and in the drawing. So, for the current matching algorithm the three drawings in Figure 11 (right) will have the same similarity value for the query in Figure 11 (left), which is not correct. In the next section we describe our new algorithm to compare geometry, which overcomes this and other drawbacks.

Fig. 10. Steps to compute the geometric similarity between the sketched query and a drawing.

Fig. 11. Query example (left) and three results with the same geometric similarity (right).
5 New Geometric Matching Algorithm

To overcome the drawbacks of the previous geometric algorithm, we focused on understanding some of the human’s visual perception characteristics. As a result of our study, we developed a new algorithm to compare the geometry between drawings, which we describe here.

In our former approach, as described above, we characterize the drawing geometry using a set of feature vectors. Each vector yields the geometric information for each given shape of the drawing. With this method, the task of determining the geometric similarity within two drawings was pretty straightforward. Therefore, if two drawings have an identical shape it was sufficient to classify them as a possible match.

Although this method proved to be effective in the content based retrieval of technical drawings, its use for sketch-based retrieval of clip-art drawings is inadequate. When a user draws a sketch to query a drawing database, he expects results to have a large number of shapes similar to the ones he drew, and not only one of them. This fact, allied to the great complexity of vector drawings led us to the development of a more robust and precise geometric matching algorithm.

Before developing the algorithm, we tried to understand how users draw sketches and what results were they expecting to be retrieved by the system. Additionally, from informal conversations with potential users of the system, we came across a few subtle details that are implicit in sketches, and therefore in users’ mind when they draw a sketch. From these information, we produced two postulates, which we express in two similarity measures: the global similarity and the partial similarity. The former compares drawings as a whole, while the latter takes into account each individual shape present in the drawing.

The first postulate is that the user normally draws a simplified sketch of what he intends to find, putting the smallest number of shapes into his query. So, at this point, the problem is on how to express this idea in terms of an algorithm that better suits users’ needs.

Therefore, and within this concept, we defined two restraints. The first one is that drawings with a number of shapes equal or superior to the query sketch are preferable. In addition, the greater the number of shapes present in a drawing, the smaller the chance to be relevant to the user’s query. The second one establishes that if the query sketch has more shapes than a given drawing, this drawing should be penalized.

So, taking these restraints into account, the global similarity of a given database
drawing is defined by the following equation:

\[ globalSimilarity = DrawingSimilarity \times Z \]

where,

\[ Z = \begin{cases} \frac{\#P_q}{\#P_{db}} & \text{if } \#P_q \leq \#P_{db}, \\ \frac{\#P_{db}}{2\#P_q - \#P_{db}} & \text{if } \#P_q > \#P_{db} \end{cases} \]

and

\( DrawingSimilarity \) - similarity between two drawings, computed using the prior geometry algorithm

\( \#P_q \) - number of polygons in the query

\( \#P_{db} \) - number of polygons in the database drawing

The second postulate is analogous to the previous one but applied to individual shapes instead of the whole drawing. It states that the user is always expecting that drawings in the result list contain the same or more instances of a particular shape specified in the query.

So, we defined two more restraints to satisfy this requirement. The first one states that drawings with an equal or superior number of a specific shape than the query sketch should be better classified. The second principle determines that drawings with a smaller number of a particular shape than the query should be penalized.

The partial similarity of a given database drawing is given by the following expression:

\[ partialSimilarity = \frac{1}{\#P_q} \times \sum_{k=1}^{\#P_q} similarity_k \times M_k \]

where

\[ M = \begin{cases} \frac{\#P_{kq}}{\#P_{kd}} & \text{if } \#P_{kq} \leq \#P_{kd}, \\ -\frac{\#P_{kq}}{\#P_{kd}} & \text{if } \#P_{kq} > \#P_{kd}, \\ -\frac{1}{\#P_{db}} & \text{if } \#P_{kd} = 0 \end{cases} \]

and
similarity\_k - similarity between shape \( k \) in the query and a shape in the drawing

\#P\_kq - number of polygons in the query similar to the current one (from the query)

\#P\_kdb - number of polygons in a drawing similar to the current one (from the query)

Combining the global and the partial similarity of the drawing we reach the following expression, that defines the total classification of a given database drawing:

\[
similarity = DrawingSimilarity \ast Z + \frac{1}{\#P_q} \ast \sum_{k=1}^{\#P_q} similarity_k \ast M_k
\]

So, drawing similarity is the sum of the overall drawing similarity with the weighted average of the similarity of each query shape.

We will now describe in detail how we determine the similarity classification of a drawing using the new geometry algorithm. To that end we will use the example from Figure 11. Considering the query with three polygons, and the first drawing in the database with two polygons, we will perform three 3-NN queries to the correspondent indexing structure. Then, and analogously to the previous matching algorithm, we build a matrix, but this time with the similarities from each polygon in the query to all polygons in the drawing, instead of distances\(^1\). The computation steps of the algorithm are illustrated in Figure 12.

Fig. 12. Steps to compute the geometric similarity between the sketched query and Drawing1 from Figure 11.

\(^1\) Regard that \( similarity = \frac{1}{e^{-dist}} \), where \( dist \) is the distance provide by the KNN algorithm.
Contrary to our prior algorithm for geometry matching, we first find the row of the matrix with the highest value (1). The following steps are equally performed until we have selected values for all polygons of the query. Finally, we sum all the selected values, and instead of simply dividing them by the number of polygons in the query, we take into account the number of elements of the query drawing and the database drawing (see equation for $globalSimilarity$ on page 15).

In this case, the query drawing has more elements than the first drawing of the database. Thus, we multiply the obtained classification by the second branch of the equation, leading us to a global classification value of 1.1. Since the two remaining drawings in the database have more elements than the query drawing, we use the first branch of the equation. Accordingly to this, the second drawing will have a global classification of 3, whereas the third one will have 2.25.

In the following step we calculate the partial similarity classification for all the polygons of the query. The first thing to do is to determine how many elements of the same type are present in the query drawing. So, we compare all the polygons between them, and the ones which have a similarity value within a certain threshold are marked as similar. In this case, the query does not have any similar polygons, since it is composed by one rectangle, one circle and one triangle. Moreover, if we take a look into the drawings in the database we notice that none of them has similar polygons. This facilitates our task, because both $P_{kq}$ and $P_{kdb}$ values are 1. Therefore, in this case, the value of $M_k$ (see equation for $partialSimilarity$ on page 15) will be the same as $Z$ from the $globalSimilarity$ equation, namely 2.2, 3 and 3 respectively.

Now, we divide the resulting value by the number of polygons in the query, obtaining a partial similarity classification of 0.73 for the first drawing and 1 for the remaining two. Finally, we sum the global similarity classification and the partial one, yielding the values presented in Figure 13.

<table>
<thead>
<tr>
<th></th>
<th>Drawing 1</th>
<th>Drawing 2</th>
<th>Drawing 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Classification</td>
<td>1.1</td>
<td>3</td>
<td>2.25</td>
</tr>
<tr>
<td>Partial Classification</td>
<td>0.73</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total Classification</td>
<td>1.83</td>
<td>4</td>
<td>3.25</td>
</tr>
</tbody>
</table>

Fig. 13. Total similarity classifications for the three drawings from Figure 11.

This method is specially concerned in retrieving drawings that have an equal
or superior number of polygons than the query drawing. In addition, it takes into account the number of elements of the same type. While the previous geometric matching algorithm classifies all the three drawings as similar to the query, our new geometric matching algorithm distinguishes them, retrieving the identical drawing as a first match. Moreover, this result is closer to the human’s visual perception.

6 Experimental Evaluation

We developed a search engine prototype for vector drawings, using our sketch-based retrieval framework and the new algorithm to perform geometric matching. The database of the system was filled with a set of SVG clip-art drawings and we carried out an experimental evaluation with users to compare the accuracy of the new algorithm against the previous one.

6.1 Indagare - A Sketch-Based Search Engine for Clip-Art Drawing

Our prototype for drawing retrieval, called Indagare (see Figure 1), supports the retrieval of SVG clip-art drawings, using sketches, an existing SVG drawing or keywords. This prototype integrates all the functionalities provided by the framework, namely, simplification mechanisms, an indexing structure to optimize the search, the topological structure to code spatial arrangement and the new geometric matching algorithm.

Fig. 14. Sketch to search for houses.
Figure 14 shows the sketch of a query, while Figure 15 presents the results returned by the implied query. If the user wants, he can submit an existing drawing in SVG format or search by keywords (input fields on top right of Figure 14). Moreover, users can also select one of the results and use it to perform query-by-example.

6.2 Experiment Description

To evaluate our new approach for geometric matching, we carried out an experiment with ten users. Six of them were male and four were female, with ages between 18 and 58 years old. None of them had previous experience with tablet devices or any other pen-based system.

Our data set of clip-art drawings is composed of 20 categories of five drawings each, selected from the OpenClipart library, yielding a total of 100 drawings.

The tests were conducted in two steps. First we asked each user to draw three sketches, using a digitizing tablet: a balloon, a car and a house, yielding a total of 30 sketches, which were stored for further use as queries. Afterwards, we show all the 100 drawings in the database and requested them to identify the drawings that they considered most similar to each of the sketches they drawn, without taking into account the drawing’s semantical value.

The second step was carried without users’ intervention. From the similar drawings selected by the participants, we identified the five more voted, and considered those as the “correct” results for each sketch.
6.3 Algorithms Evaluation

To evaluate the new algorithm for geometric classification we submitted the three sketches from each participant to the system and collected the results. We configured the system to retrieve 30 results for each query. With these results we computed precision and recall values.

6.3.1 Using just Geometry

In this experimental test, our primary concern was to analyze the new geometry algorithm effectiveness. Therefore, our first step was the comparison between the old geometry algorithm and the new one, without the topology comparison step. We calculated precision & recall levels for the two methods, using the 11-Point Interpolated Average Precision method. The mean precision for each recall value of the two algorithms is presented in Figure 16.

Comparing the precision at each recall level, we can observe that adding implicit information, like the number of shapes in a drawing to the geometric similarity calculation, substantially improved the algorithm. We can also notice that with the new geometry algorithm at the 10% and 20% recall levels the precision suffers a boost of 54%. Furthermore, fixing the average precision at each recall level of the new geometry algorithm, we realize that its precision is only overtaken at the 90% and 100% recall levels by the precision of the old geometry algorithm at the 10% and 20% recall levels. Therefore, and since the 90% and 100% recall levels correspond to the retrieval of the 5 relevant documents, we can conclude that, in average, for every 4 relevant documents retrieved by the new geometry algorithm, the old geometry algorithm retrieves only 1 (Figure 16). So, according to the current results, the refinement made in the geometry algorithm produced undeniable improvements in the system’s

![Fig. 16. Precision and recall for the two geometric matching algorithms alone.](image-url)
effectiveness.

6.3.2 Using Topology and Geometry

Our next step was to evaluate both geometric algorithms in combination with the topology filtering. From our prior evaluation of the original geometric matching algorithm [3] we concluded that the best configuration for it is to first compare by topology and then compare by geometry.

Figure 17 presents the results for both geometry algorithms when we first perform the topology filtering. As we can observe, the new method presents better results than the old algorithm. However, the achieved precision is smaller than that obtained with the new geometric matching alone (see Figure 16).

Taking these results into account, and recalling the previous observation stating that clip-art drawings are more geometric than topological, we decided to perform first a geometric filtering and then the topological comparison. We did this only for the new algorithm, because this combination was already evaluated for the old one and did not produce good results.

Results comparing this new sequence with the best combination for the old geometric matching algorithm is depicted in Figure 18. From this chart, we can identify that comparing first by geometry and then by topology is the matching solution that achieves better precision values. Comparing it with the results achieved by the first method (topology plus old geometry algorithm), show an improvement superior to 50%, at the 10% and 20% recall levels.

So, at this point we have two possible solutions for the matching process that produce the best results. One using just the new geometric algorithm and another that uses the new geometric algorithm plus topology.

![Figure 17: Precision and recall for the two geometric algorithms, when we first compare by topology.](image-url)
Fig. 18. Precision and recall for topology plus the old geometry algorithm and for the new geometry algorithm plus topology.

Figure 19 presents a comparison between these two. As we can see, the new geometry algorithm plus topology configuration is slightly better than the new geometry alone. In fact, it presents superior results at 10% to 40% recall levels and inferior values for 70% to 100% recall levels.

So, we need to check if the improvements in the final results, caused by the inclusion of another step in the matching process, compensates the additional time spent on it. To answer this, we carried out a new test to measure the time needed to execute a query on each algorithm.

Fig. 19. Precision and recall for the two best combinations for the matching approach.
6.4 Time Evaluation

To collect query times for both methods, we submitted ten queries to the two different configurations, using six databases of 100, 500, 1000, 2000, 5000 and 10000 drawings. We measured the time taken by the system to retrieve the 30 more similar documents to the query\(^2\). Results are shown in Figure 20.

![Time taken to perform a query for the two best configurations](image)

From the chart, we notice that in the worst case the configuration that uses both comparison stages takes more six seconds to perform a query to the database of 10,000 drawings than the other.

According to these results (see Figures 19 and 20) we can configure the matching process for two different types of users: One for users that want more accurate results disregarding the time they take to be generated. In this case we should use both Geometry and Topology comparison stages. Another for most common users that are more concerned in getting good results in a short period of time.

6.5 Large Databases

After measuring the time needed to perform queries, our next step was to study the behavior of our new algorithm with large databases. To that end, we considered two databases, one with 2,000 and another with 10,000 clip-art drawings. These databases included the first set of 100 well know drawings.

In these tests we used the same set of sketches drawn by users as queries. Due to the large number of drawings into the databases, it is almost impossible to

\(^2\) Tests performed on a Pentium Core 2 Duo 2,0GHz, with 2GB RAM, running Windows XP.
Fig. 21. Precision @ N for the three algorithms, using a database of 2,000 (left) and 10,000 (right) drawings, using relevant results identified by users.

know every drawing inside them and consequently we could not know what would be the relevant answers to a submitted query. That is, we can evaluate the precision of the returned results, but we can not measure the recall. So, instead of using Precision & Recall measures, as we did in the previous tests, here we use the Precision @ N, which gives us the precision value for a specific number of results.

In the first test, with the large databases, we compared the new Geometry algorithm (with and without Topology) against the previous matching algorithm (Topology + Old Geometry). To that end we considered the set of relevant results identified by users as the correct answers. Figure 21 shows the results for the three algorithms using the 2,000 (left) and 10,000 (right) databases.

As we can see, the new geometry matching algorithm presents better results than the old algorithm, for both databases. However, precision values are very low (around 1.5%).

During experimental evaluation, we notice that there were “new drawings” returned by the system, very similar to the queries, but not included in the initial set of relevant drawings identified by users. Indeed, by inserting more 2,000 or 10,000 drawings in a database, the likelihood of having more draw-

Fig. 22. Precision @ N for the three algorithms, using a database of 2,000 (left) and 10,000 (right) drawings, using all relevant results.
ings similar to our query is higher. To solve this, we decided to consider as correct results not only the initial set of relevant results, but also all drawings similar to the query. Figure 22 shows the results for the two databases. From the figure, we can see that the values of Precision @ N increase significantly, achieving more acceptable values. For a database of 2,000 elements we reach precision values of 20% while for 10,000 we achieve 10%.

Finally, Figure 23 shows the behavior of our new algorithm when the size of the database increases. As we can see, our new algorithm (in both configurations) keeps almost the same precision value for the two databases, 100 and 2,000 drawings. This is a good result, revealing that our solution can be used in real databases with thousands of drawings.

![Fig. 23. Precision @ N for the two configurations of the new algorithm for a database of 100 and 2,000 drawings, using all relevant results.](image)

Although, the precision @ N values for our approach seem low (around 20% for 2,000 drawings), it is important to emphasize that we are doing retrieval of complex drawings using simple sketches as queries, while most related works retrieve simple drawings using query-by-example. For instance Leung and Chen [11] use a small database of hundreds of simple sketches, performing query-by-example and using the elements in the database as queries. Namboodiri and Jain [13] evaluated their approach using a database of 150 very simple sketches by comparing each of the sketches against every other sketch in the database. Pu and Ramani [17] compare sketches with drawings, and use a database of 2,000 drawings. However, their drawings are not so complex as clip-arts and their values for precision & recall are only slightly better than ours.

7 Conclusions and Future Work

In this paper, we proposed a new geometric algorithm that uses the information about the number of visual elements in the query and in the database
drawings to compute a more accurate measure of geometric similarity. This new algorithm was integrated into our framework for sketch-based retrieval of drawings, and its efficiency was compared to the prior geometric algorithm. Experimental evaluation showed that the new algorithm outperforms the previous one, providing an increase of 50% in the retrieval effectiveness of the system.

Although the most efficient solution is the geometry filtering followed by the topology matching, it is very time consuming. So, the solution that provides a better tradeoff between quality of results and processing time, is the one that uses the new geometric matching algorithm alone.

Currently, the creation and organization of the information extracted from drawings is optimized to use topology as the first filter. So, we believe that the overall querying time can be reduced if we adapt/redesign the framework to first filter by geometry and then by topology.

To further enhance the system precision, we plan to use topological information in the computation of the geometric similarity. To that end, we could assign weights to shapes according to its importance in the drawing. For instance, a drawing containing a shape that includes a great number of other shapes, should be more relevant than a drawing that has a shape containing less polygons.

References


