6th International Conference on Applied Informatics
Eger, Hungary, January 27–31, 2004.

On the Visualization of Image Databases

Krisztián Veréb

Department of Information Technology, Institute of Informatics,
University of Debrecen
e-mail: sparrow@inf.unideb.hu

Abstract

There is an approach in [6] about spaces in image databases. It distinguishes five spaces, namely the composition space C, the query space Q, the feature vector space F, the output space O and the display space D. In case of a general query in an image database, users formalize their query in space Q via space C to transform space F into space O. These steps are very important to get the similar images to a query image q as a result set in the image database. But it is very important to display these images as well. This step — namely the visualization — can be considered as a transformation between O and D. In this paper I show four different model to visualize the result sets. The models are different in some features, e.g., the navigation and the computational time. One of the models is a self-developed one, called star model. It tries to give a good characteristic in the above mentioned two features. The advantages and disadvantages of the models are also presented in this paper.

1. Introduction to Image Databases

First let us see why storing of images may be necessary besides textual information (you can see some particular applications in [1]). One reason is when image information has a complementary role. In the other group there are such cases when the image is not only a complementary information, but it is the subject of the archive itself. The aim of the database is the storage of images. So in this case images are not only sequences of bits but also complex information, so we know the content of images. The third group is the rest, where there is no textual information only images.

Until now we have not mentioned the retrieval (see [7] for more details). In the first group the direction of retrieval is trivial — the input is the textual description, and the output is the image. It is the same in case of the second group. But it is possible to use the opposite direction — the input is the image, the output is the textual information. Hence the claim of content-based information retrieval has

appeared. In case of the last group there is no textual information, the image itself is to be gained. In this case the input can be a part or parts of images, or a poor quality version of the needed image. So, the input and the output are images as well. The relationship between them is the similarity.

If complementary description connects to images, a direct keyword-based retrieval can be executed. If the textual description is not available, the content-based image retrieval has to be used. These approaches are based on features extracted directly from the raw image, e.g., color, shape, texture. The usage of the main visual information retrieval paradigms can be divided into three classes. They are similar-picture queries [5], sketch-based queries [2] [4], and icon-based queries [3]. In the first class the user gives an input image, and the system returns a sequence of images similar to the input. In the second class the user draws an image skeleton, which is the input of the query. In case of the iconic retrieval, the user places different icons in different image points defining what features are important in each location. Applying any of these approaches, the method is to define an input image, which can contain additional information, e.g., applicable matching strategy etc. — and the output retrieved by the system is a collection of images meeting the given matching criteria (e.g., the given similarity levels).

To compare two images, beyond the pixel-by-pixel pattern matching, several refined methods can be used. The existing techniques are identical in comparing feature vectors extracted from images instead of comparing whole images [3]. The feature vectors can be selected into two groups [6]. The first group is where images can be reconstructed from the vectors. It is called representation. In the other group there are vectors which the image cannot be restored from. These vectors can represent some measureable feature of the image or the contained objects. They are called characteristics. To decide the similarity, the matchings often need both of them. The word "feature vector" denotes the vectors of these classes.

Let us take a closer look at the process of a general query. The user defines the query image by the help of an (similar-picture, sketch-based or iconic) interface. The output is a set of images similar to the query image. Let the query be denoted by q. Then the system extracts the feature vectors of q, matches them to the vectors of the images in the database. The result set has three groups. The first is when only one exact matched image and the additional information are to be searched for. In this case the identical query can be used. The result set O (or by an other terminology the output space) contains images f where

$$d(f, q) = 0.$$

The d is the virtual distance of the vectors. This is a measuring number expressing the similarity. An environment ϵ can be given as well (threshold). Thus those pictures f are searched where

$$d(f, q) < \epsilon.$$

This is the ϵ-query. The searching for the nearest neighbours (NN) is in the third group. The result set is

$$\forall p, p \neq f, d(f, q) \leq d(p, q).$$

It is very familiar to use similarity measure instead of distance. In other words, there is a similarity measure S depending on the distance d, where $S \in [0, 1]$ is a real number. Its value is 1 if and only if d is zero.

2. About Spaces in Image Databases

There is a classification in [6], grouping the spaces of the database in the following way. As we mentioned before, the queries of the images are based on their feature vectors. Hence, the first space of the image database is the feature vector space, denoted by F. In real, this space is a composite space of several different feature vector spaces. It comes from that different matching algorithms use different feature vectors. Several metrics can be defined in space F ensuring the basis of the matching algorithms.

When a query is given, we can give a query image q. In this query an output space O has to be transformed for q from space F. In fact this is the space of the resulted images. Of course it depends on space F and the facility of the searching and last but not least it depends on the query image q itself. In the most general case, space O is the same as space F according to their elements except their distances. In this case the distances are different, so the "ordering" of the elements is changed.

When space O is given, then the images (or a subset of the images) of space O have to be displayed. In other words, space O has to be transformed into a display space D, i.e., onto the screen (or printer etc.). That is a user friendly representation of a subset of O. The user can contact the elements of the output space O through this display space D.

In case of general queries there are two other spaces to be used as well. The first is the composition space C. That is the user interface which is used to provide a particular query. This space can be icon-based or sketch-based, etc., in accordance to the first section. In real, the given query is built up in the query space Q, but space C has to be used to bridge the gap between the user and Q (analogously as space D for O). In general, D and C are some projection of spaces O and Q. So the parameters, the used matching algorithms, the used distances and the weighting techniques have to be defined in space Q using space C.

So, as a summary, a scenario of a general query is the following. The user — using space C — gives a query image q and a query in Q. This query transforms space F into space O. Space O has to be displayed via D. Spaces C and D are in the screen, Q is the space of q (with their feature vectors and matching algorithms). Space F is the space of feature vectors of the stored images with various distances d. Space O contains elements of F with a similarity measure S depending on q and Q (and d of course).

The structure of space C, i.e., the structure of query interfaces has a large literature (see section 1). In the next session let us see in more details how a query can be built up in Q and how it trnasforms F into O.

3. On Getting the Result Set

Let us consider an image database having n matching algorithms $(Q_1, \ldots, Q_n)$. Let us assume that in case of matching images q and f the algorithms result similarity measures $Q_i(q, f) \in [0, 1]$, i.e., $Q_i(q, f) = 1$ if and only if q and f are identical according to the given algorithm, and its value is zero if they are totally different $(i = 1, \ldots, n)$.

Thus — in case of a given query image q and a stored matched image f — one can get matching results $Q_1(q, f), \ldots, Q_n(q, f)$. Most of the systems combine these results to get a total (global) similarity measure. It is rather familiar to use the weighted sum, where the user has to give a real weight vector $\underline{w} = (w_1, \ldots, w_n)$, $w_i \in [0, 1]$, $(i = 1, \ldots, n)$, where

$$\sum_{i=1}^{n} w_i = 1.$$

The weights in this formulae mean the relevance of the features. Thus the total similarity value S for an image f in case of query q can be derived in the form of

$$S(q, f) = \sum_{i=1}^{n} w_i Q_i(q, f).$$

If the database contains $f_1, \ldots, f_m$ matched images, then in case of a particular query image q the matching matrix is

$$\boldsymbol{A}(q) = (Q_i(q, f_j))_{n \times m},$$

where $j = 1, \ldots, m$ and $i = 1, \ldots, n$. In this case one can get the vector $\underline{r}(q) = (S(q, f_1), \ldots, S(q, f_m))$ for images m, which can be defined as

$$\underline{r}(q) = \underline{w}\boldsymbol{A}(q)$$

Let us denote the elements of vector $\underline{r}(q)$ by $r(q)_i$ $(i = 1, \ldots, m)$ in the followings.

Because m (the number of candidate images for matching) is often large, so the system will not return all of the images only a subset of them in general. If only the image or images are searched with maximal similarity value, then the result set O is

$$O(q) = \{f_i | \max_i r(q)_i\}.$$

But in this case — because of the characteristics of the matching algorithms — it is often impossible to find the searched images. That is the reason why the user has to give a real treshold t, $t \in [0, 1]$, which defines an error to find the most similar images. If $t = 0$, then all of the stored images will build up the output space. If $t = 1$, then we are looking for the totally identical images. So the result set O contains those images f_j, where the matching value is greater than the treshold, so

$$O(q, t) = \{f_j | r(q)_i \geq t, j = 1, \ldots, m\}.$$

Note, that there can be given a one-to-one correspondence between the treshold t and the mentioned environment ϵ.

4. Projection Between O and D, Visualization and Navigation

At this point, the question is how the multidimensional space O can be displayed in a 2D screen, i.e., how space O can be transformed into space D. One of the most familiar models — which needs a given treshold t and a particular weight vector $\underline{w}$ — is the line model. With given q, t and $\underline{w}$ the output space O can easily be defined, and one can sort of elements f_j into a line ordering their values $r(q)_j$ by descending. In this case the navigation is only a walk from q towards the least similar images. This model is not a space effective model, but it can be applied well if extra information — e.g., textual information — is needed to display with the resulted images as well (see figure 1).

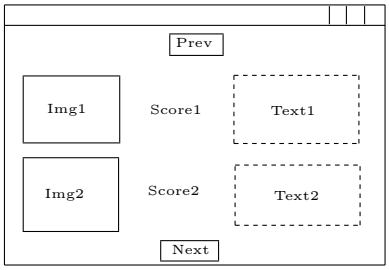

Figure 1: Display with line model.

If there is no treshold t given, it is recommended to avoid displaying all images of O. It is very familiar to put the images into a matrix $\boldsymbol{M} = (m_{i,j})_{3\times 3}$, which contains those 9 images f_k where their values $r(q)_k$ are greater than the others' values. It is often called matrix model. Here the navigation is a paging from the first 9 images f (including q) step by matrices towards the less similar images in a sequence of matrices $\boldsymbol{M}_l$ $l = 1, 2, \ldots$. Every next matrix contain images f with the next 9 greatest values r. It is similar to the line model (nearly equivalent), but it has better space improvement. Unfortunately because of the good space improvement, extra textual information cannot be displayed near the images in general (see figure 2).

The previously mentioned two models are common in requiring vector $\underline{w}$ and in the transformation space O into a line which has the endpoint q. If weight vector $\underline{w}$ cannot be given, the question can be reformulated as how the nD space can be projected into a 2D space. The visualization has to restore the similarity of images f_j and q, but it is a very important suggestion, that this projection ahould not be too difficult to compute and it has to provide some navigation feature as well.

Such a model is the fish-eye model. Actually, this model is to visualize the 2D space better (i.e., it depends on the nD-2D transformation π in large extent). If the space is transformed into 2D where the origo is q, then it can be placed into

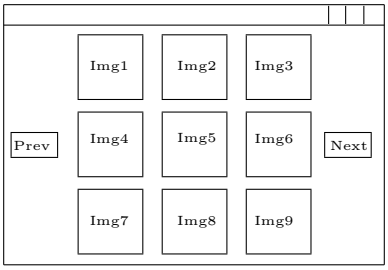

Figure 2: Display with matrix model.

the centre of the display, and the 2D space can be streched onto a hemisphere, thus the whole infinite 2D space can be displayed in a finite screen. In 1D the display can be seen in the figure 3.

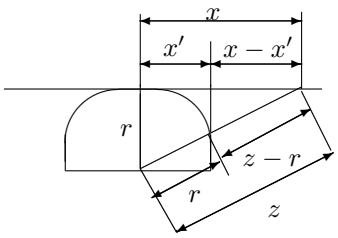

Figure 3: One dimension fish-eye display.

It can be seen, that a point x' suitable for x depending on r can be defined from

$$\frac{1}{x'} = \frac{1}{x^2} + \frac{1}{r^2}$$

easily. The solution is perfect in the sense of the navigation, i.e., we only need to "turn" the projected space on the surface of the hemisphere. In other words, we have to declare an image f_i as origo. The model also has disadvantages, e.g., it depends on the projection π in large measure, and the navigation may not express the feature differences in some cases. It can be stated, that every pixel of the images f has to be projected to the hemisphere, so it is very computational-intensive. To avoid this lack I have developed the following model.

5. The Star Model

In case of star model space D is built up in the following way. In the screen at the same time $2n+1$ images can be seen. There is an important (emphasized) location, called origo, and there are $2n$ nearest positions $m_i^+, \ldots, m_n^+$ and $m_0^-, \ldots, m_n^+$ according to the n matching algorithms. The current image in the navigation are in the origo, and in positions m_i^+, m_i^- are images closer $(+)$ or further $(-)$ to the q respectively to the i^{th} feature. So, there is n ordered sequence, i.e., the matrix $\boldsymbol{A}(q)$'s i^{th} row has to be sorted to determine the ordering of images f_j. Let us denote the ordered sequence of f_j, sorted by the i^{th} row of the matrix $\boldsymbol{A}(q)$, by $f_j^{'(i)}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. In each sequence there is an image f_j only once. For each sequence there is an endpoint q, which has no previous element, i.e., $f_0^{'(i)} = q$, $i = 1, \ldots, n$.

Thus the navigation is the following: choose a feature i, and step to the previous or the next element (towards or backwards q), hence an image f_k will be placed into the origo. Then choose an other feature, and step again, thus an other f_k will be the origo, etc. In figure 4 a star model display can be seen.

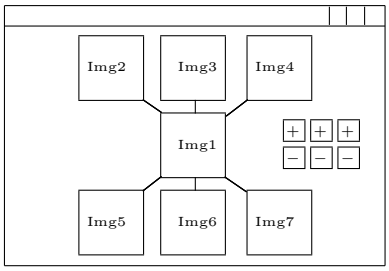

Figure 4: Display with star model in case of three features.

In a general case if there is an image f_k in the origo, which image is in the i^{th} sequence at the position k_i, and we step in the sequence i towards the q, then the element $f_{k_i-1}^{'(i)}$ will be placed into the origo. In the n sequence this image is in positions j_l respectively $(l = 1, \ldots, m)$. Thus images $f_{j_l+1}^{'(l)}$ will be placed into positions m_l^-, $l = 1, \ldots, n$, and analogously images $f_{j_l-1}^{'(l)}$ will be in m_l^+, $l = 1, \ldots, n$ respectively. If there are no such images (because the particular $f_{j_l}^{'(l)}$ was an endpoint in some l^{th} sequence), then an empty image will be placed into the position in question. (For example, if the query image q is in the origo, then all of the positions m_l^+ are empty.)

The name of the model "star" comes from the figure drawn by the image positions on the screen.

In case of this model the navigation can express the differences in features, it is not computational-intensive, and there is no need to project the nD space into a 2D

one. But it has a disadvantage, namely it cannot indicate the clusters (groupings) in space O. But nevertheless in case of a satisfying projection π the fish-eye model is suitable to indicate such clusters.

6. Summary

In the previous sections four models were introduced for the visualization of space O in D. The models provide some navigation features in space O. Actually, none of them is better than the other. They are alternative solutions to visualize the images of the result sets in image databases. They are different in various parts of the visualization and the navigation. There are parts where they are great and in some other problems they are not too good.

The line model is great when textual or extra information is needed to be displayed with the images, but it is not too strong in the navigation, and it can display only a few images at the same time. The matrix model increases the number of the commonly displayed images, but its navigation is also weak, and cannot give good solution to display extra or textual information. At this point, we want to improve the navigation facility of the models. Therefore the fish-eye model is proposed to use in [6]. It is great in the navigation and it can indicate the clusters in space O, but it is very computational-intensive. That is the reason why I developed the star model, which has good navigation feature, it does not decrease the number of the displayed images at the same time and, mainly, it is not computational-intensive. Unfortunately, it cannot indicate the clusters in O, and cannot give a good solution to display extra information with the images (neither can the fish-eye model).

In the following table we try to show the differences between the four models:

Model	**A**	**B**	**C**	**D**	**E**
Line model	−	−	+	−	+
Matrix model	+	−	−	−	+
Fish-eye model	+	+	−	+	−
Star model	+	+	−	−	+

where column **A** means that the models can display more images at the same time, column **B** stands for the models' good navigation features, column **C** means that the models can display extra information, column **D** means that the models can indicate the clusters in O, and column **E** stands for the models' easy computational facility.

References

[1] V. Castelli, L. D. Bergman (ed), Image Databases, Wiley, (2002)

[2] M. Egenhofer, Spatial-Query-by-Sketch, VL'96, IEEE Symposium on Visual Languages, (1996), 60–67

[3] M. S. Lew (ed), Principles of Visual Information Retrieval, Springer, (2001)

[4] S. Matusiak, M. Daoundi, T. Blu, O. Avaro, Sketch-Based Images Database Retrieval, MIS'98, LNCS 1508, (1998), 185–191

[5] D. Papadias, T. Sellis, A Pictorial Query-By-Example Language, Journal of Visual Languages and Computing, 6(1), (1995), 53–72

[6] S. Santini, Exploratory Image Databases, Content-Based retrieval, Academic Press, (2001)

[7] B. Thuraisingham, Managing and Mining Multimedia Databases, CRC Press, (2001)

Postal addresses

Krisztián Veréb
Department of Information Technology
Institute of Informatics
University of Debrecen
H-4010 Debrecen, P.O.B. 12.
Hungary