

# PROFI



Project number:	FP6-511572
Project acronym:	PROFI
Project Title:	Perceptually-Relevant Retrieval of Figurative Images

Deliverable No: D8.2
----------------------

Title: Experimental results of Bayesian relaxation
--

## Short description:

During this stage, the graph-matching framework to identify similar images using components will be designed. Inside the proposed graph, each graph node will represent an image component (edge segment or shape) and the graph edges will represent the topological and directional features between components. The matching framework will be designed using the Bayesian version of the Relaxation by Elimination framework (Turner and Austin, 1999). Partner 2 (UoY) possess extensive experience in applying this framework to several problems including chemical structure matching and trademark search.

This framework consists of two main stages: making initial hypotheses about possible matches; improving initial hypotheses using contextual information. During this work, the required methods to perform these tasks, within the scope of the current context, will be designed. UoY will supply a binary executable of the Relaxation by Elimination algorithm. It will also share its experience in using this algorithm to solve similar problems.

Due month:	33
Delivery month:	33
Partners owning:	UoY
Partners contributed:	UoY
Classification:	Restricted



Project funded by the European Community under the  
“Information Society Technologies” Programme

# PROFI

Perceptually-Relevant Retrieval of Figurative Images

Deliverable 8.2

Experimental results of Bayesian relaxation

Design the graph matching framework to identify similar images

**Victoria Hodge, Mike Weeks, John Eakins  
and Jim Austin**

Advanced Computer Architectures Group  
Department of Computer Science  
University of York, York, UK

29th February 2008

## Table of Contents

Work package 8 Objective 2.....	4
Overview.....	5
1 Introduction.....	5
2 View generation and shape identification.....	9
2.1 Scale Space representation.....	9
2.2 Gaussian Pyramids.....	9
2.3 Categorisation.....	10
2.4 View Generation.....	11
2.4.1 For region/line-based images.....	12
2.4.2 For texture/noisy images.....	12
2.5 Results for view generation.....	13
2.6 Shape Identification.....	14
2.6.1 Similarity Pruning.....	18
2.7 Results for perceptual shape finding.....	18
3 The Graph Matching Framework.....	20
4 Evaluation.....	26
4.1 Data.....	26
4.2 Evaluation Framework.....	27
4.2.1 Score measures.....	34
5 Results.....	36
6 Conclusion.....	40
References.....	42
Appendix A.....	44
7 File Formats in use by the Trademark Matcher.....	44
7.1 Database Files.....	44
7.1.1 imageX_local_raw.....	44
7.1.2 imageX_global_raw.....	45
Appendix B.....	47
8 The unprocessed images for Query 23285. (Query with lowest normalised recall).....	47
Appendix C.....	55
9 The unprocessed images for Query 23300. (Query with highest normalised recall).....	55

## Work package 8 Objective 2

The following is the original objective of Work Package 8.2 as given in the project proposal document:

During this stage, the graph-matching framework to identify similar images using components will be designed. Inside the proposed graph, each node will represent a node and the edges will represent the topological and directional features between components. The matching framework will be designed using the Bayesian version of the Relaxation by Elimination framework (Turner and Austin, 1999). Partner 2 (UoY) possess extensive experience in applying this framework to several problems including chemical structure matching and trademark search.

This framework consists of two main stages: making initial hypotheses about possible matches; improving initial hypotheses using contextual information. During this work, the required methods to perform these tasks, within the scope of the current context, will be designed. UoY will supply a binary executable of the Relaxation by Elimination algorithm. It will also share its experience in using this algorithm to solve similar problems.

***Deliverables:*** D8.2 A report explaining the results of the experiments carried out to assess the retrieval accuracy of the new methods.

## Overview

In this report we focus on identifying similar trademark images using a graph matching framework. The pre-processing task is to produce the data to represent each image to allow matching. The data (images) must then be matched and, for a particular query image, the most similar images retrieved using a graph matching framework. The report is self contained, in that it describes both the pre-processing used as well as the graph matcher. The pre-processing systems are shared with the other parts of the PROFI project.

## 1 Introduction

There has recently been massive growth in the area of digital image processing. Examples of image processing applications include archiving photographs, medical image analysis and trademark retrieval. Content-based Image Retrieval (CBIR) is a sub-area of digital image processing where the aim is to retrieve images from an image database that are similar to a query image. CBIR takes a query image and attempts to find all matching images: images which might be deemed similar to the query image by a human analyst. Trademark image searching and retrieval is potentially one of the most important application areas for CBIR techniques. Trademarks have major commercial significance and form an important part of any company's intellectual property. There are many large and growing databases of trademark images. These large databases need accurate and fast indexing and retrieval systems to allow the trademark databases to be searched effectively. New trademarks need to be matched against the existing database to ensure there is no copyright conflict. Searching for potentially conflicting trademarks among databases comprising solely of image data is a difficult task [E00]. Different humans perceive images differently [HEA06, HHEA06] and the trademark examiner has to judge whether any feature of the query trademark could reasonably cause a member of the general public to confuse it with an existing trademark.

*In this report, we focus on the task of using computerised methods to match and retrieve trademark images to produce a system that emulates human matching.* Trademark systems generally represent the image as a set of features which are stored in the image database along with the images. Query images are then matched against the database of stored images using feature matching. To allow matching and retrieval we need to find a suitable image representation format. This encompasses both the format of the data and the way that data is represented.

Images may be represented globally (holistically) or split into components and represented either as a collection of components, or as a set of individual components that may be matched individually. There is also the question of the format of the data. There are various strategies adopted in the literature for both the data and the organisation of the data. Kim and Kim [KK98] have adopted a global approach using Zernike moments to capture the global properties of images. They use the distribution of moments across the entire database to identify the significant features (moments) and then use these extracted moments to calculate image similarity. Use of deformable image templates is another global approach sometimes used [DP97]. The query image template is deformed to match with another image in the database. Similarity is calculated using the similarity between two models after deformation and is expressed as an energy function. However, the energy required to transform the template depends on the complexity of the image and thus neglects the qualitative features of images. In Ciocca and Schettini's work [CS99] moments, a histogram of edge directions and wavelet coefficients form the features. These global approaches compute the *global* similarity between the query and the database trademark so, the retrieval system will not be robust if the query or database images consist of different numbers of shapes as the number of shapes will affect the matching score. Local approaches include

Kato's trademark retrieval system which employs a coarse local approach. It divides the images into a series of square blocks (e.g. 8x8 pixels) and calculates the features of each of the blocks. However, this ignores the edges and boundaries inherent in the image and is too coarse to give the desired accuracy. Eakins et al. [ESB96] introduce the ARTISAN system based on principles deriving from Gestalt psychology. The first system [ESB96] extracts the boundaries of the components of a trademark and groups similar components into families. Image similarity is calculated using the distance between feature vectors. Subsequent ARTISAN versions [ERE03] show improvements in the grouping phase, the use of multi-resolution analysis to cope with texture and noise, and the introduction of a variety of shape measures. However, the spatial relation between components within the same image is not considered. We follow the theory of using a multi-resolution approach as this agrees with the psychology of human matching (discussed next) but we also intend to incorporate a degree of shape relations into our system. Jain and Vailaya [JV98] introduce a two phase system that uses moments and edge histograms in the initial global phase to find candidate matches. There, the candidates are refined during the subsequent matching phase using deformable template matching to compare the edge map of the query trademark and the edge map of the candidate trademarks from the initial phase. We also use a two phase process here but our two phases are focused on processing the image to produce multiple perceptual representations and then matching these image representations using a graph matching framework. Manmatha, Ravela and Chitti [MRC98] use both global and local features for trademark image retrieval. They aim to capture both local and global similarity of images using features that describe different levels (resolutions) of images. The use of features from different levels of images is an attractive feature of this system and we aim to incorporate this multi-level approach into our system to allow images to be matched at different representational levels.

Most experts agree that shape similarity is the most important determining factor for trademark image similarity in humans [E00]. However, in humans, image similarity is not just determined by the similarity of simple image shapes but also encompasses higher-level patterns (structures) made by the individual shapes following the Gestalt principles such as similarity, proximity or continuity [G72]. In Figure 1(a), the texture can effectively be ignored and treated as a solid area so that to a human, the two triangles appear similar while in Figure 1(b) the image components are structured at a higher level as a ring-shape so the humans view the three ring structures as similar even though the individual components used to produce each of the three ring structure are quite different.



**Figure 1. Examples of trademark groups considered similar: (a) presence or absence of texture (b) similar overall image structure (ring-shape).**

The principal aims of our computerised image matching and retrieval system are therefore: producing image views with perceptual significance (similar to the views perceived by humans [HEA06, HHEA06]) and mimicking Gestalt processing to match images. Thus, we introduce an approach for matching and retrieving images which comprises two phases: 1) finding patterns (structures and shapes) in trademark images at different perceptual levels emulating the Gestalt principles followed by 2) matching images at different perceptual levels using sub-graph isomorphism, where the shapes from the first phase form the data to represent the images.

In the first phase we would, for example, aim to represent both shapes in Figure 1(a) as triangles and both shapes in Figure 1(b) as ring structures. The Gestalt principles refer to the shape-forming capability of human vision. In particular, they refer to the visual recognition of structures and whole shapes rather than just 'seeing' a simple collection of lines and curves as we demonstrated in Figure 1. Hence a computerised image retrieval system must be able to identify and match the most salient aspects of an image's appearance including: the image's overall shape; the shapes of important image components; or, shapes defined by perceptually significant groupings of components.

Finding perceptual structures and shapes requires generating image representations (views) at different levels. This is a difficult task that requires a "semantic" level of understanding and a number of different processing methods as no one technique is universally effective. By integrating a series of techniques, we aim to overcome the limitations of each individual technique while exploiting their strengths. In IBM's QBIC system [A95] each image in the database has multiple representations achieved through the use of different feature spaces of an image rather than by generating new views at different scales. French et al. [F03] introduce an image retrieval system that employs multiple image representations using transformations of the different colour channels and then consolidates the results of matching the different representations to produce a ranked list of results. We take our cue from French et al. [F03] and generate multiple views of the image although here we are only interested in black/white or greyscale images. We use scale space selection [L98] and Gaussian pyramids [BA83] to blur the image followed by pixel clustering to extract the image structures at different levels and produce multiple representations of the image (multiple **image views**). After clustering, we identify the shapes and structures within the image views using edge segmentation and linking that obeys the Gestalt principles of continuity and proximity. This produces a set of **image views** for each image and each view has a set of **edge segments** and a set of **shapes**. These sets represent the edge segments and shapes present in the image at different perceptual levels; they form the data representing the images which will be used by the graph matching framework to match and retrieve images from the image database.

The second phase of the system aims to match and retrieve similar images. A global (feature based) approach is sensitive to the number of components in the image which is undesirable, and trademark images vary widely in the number of components. Therefore, we adopt a local, geometric based, approach. As stated earlier, our aim is to mimic human similarity matching and the key component of image similarity for humans is shape similarity. Therefore, our data comprises the shapes extracted from the first phase, and the relationships between the edge segments from the first phase in a two level system. To identify image similarity we use an evidence counting approach, coupled with graph matching (employing the relaxation by elimination (RBE) technique). We match images at two levels: using primitives and using shapes/structures. We represent our primitives by using the perceptual edge segments and their respective relationships that were identified in the first phase. These segments form the graph nodes and the three relationships: collinearity, proximity and corner represent the graph arcs. A pair of nodes is connected by one or more appropriate arcs if the corresponding segments are related (i.e. collinear, proximal or if they form a corner). For each shape/structure extracted from an image in

the first phase, we match shapes using vector similarity within the graph matcher. Shape matching also uses RBE and incorporates the perceptual neighbourhood of the shape and evidence percolated up from the primitive (edge segment) layer. This approach has been shown to outperform conventional Euclidean distance scoring using feature vectors [A99]. The strength of match between two images is then calculated by combining the evidence from primitive (edge segment) matching and higher-level (shape/structure) matching to produce a single image view score with which the image views in the database may be ranked.

The following sections describe our approach in detail. Section 2 provides an overview of the image processing techniques we use: 2.1 Scale space selection to smooth (blur) images at appropriate kernel sizes (scales), 2.2 Gaussian Pyramids to form a pre-processing step for scale space selection to introduce systematic pre-smoothing of the image, 2.3 Categorisation to identify the regions present in the image using the image's pixel intensities. These we combine to produce our overall methodology which is detailed in section 2.4. In section 2.5 we describe edge detection which allows us to outline and identify the image structures in the image views we have produced. In section 3, we describe the graph matching framework in detail. Section 4 provides details of our recall and precision analyses on a series of trademark images and an analysis of the results produced using our processing technique is provided in section 5. Section 6 is the conclusion.



## 2 View generation and shape identification

Sections 2.1-2.4 describe how we merge lower level shapes and texture within the image to extract structures and produce perceptual views of the image. Section 2.5 describes a shape identification algorithm to determine the shapes present in these views and to identify other perceptual structures missed by the view generation step.

### 2.1 Scale Space representation

The first step for generating multiple perceptual views is image scaling. Scaling an image by different amounts allows us to identify different levels of structure within the image by blurring (merging) lower level structures and thus revealing the higher level structures, for example removing texture and grouping shapes. Here we develop the scale-space method of Lindeberg [L98] which automatically selects the optimum scaling factor.

The *scale-space representation* for a 512x512 pixel 2-D image ( $I \in \mathfrak{R}^2$ ) of continuous  $f: \mathfrak{R}^2 \rightarrow \mathfrak{R}$  where  $f(x, y)$  is the pixel intensity at  $(x, y)$  is  $L: \mathfrak{R}^2 \times \mathfrak{R}_+ \rightarrow \mathfrak{R}$  which is given by the solution of the diffusion eqs 1 and 2.

$$\partial_t L = \frac{1}{2} \nabla^2 L = \frac{1}{2} \sum_{i=1}^D \partial_{x_i}^2 L \quad \text{with } L(\mathbf{x}, 0) = f(\mathbf{x}) \quad \text{where } \mathbf{x} = (x, y) \in \mathfrak{R}^2 \quad (1)$$

$$L(\mathbf{x}, t) = (g(\cdot, t) * f(\cdot))(\mathbf{x}) \quad \text{where } g(\mathbf{x}, t) = \frac{1}{(2\pi)^{D/2}} e^{-\mathbf{x}^T \mathbf{x} / (2t)} \quad \text{and } * \text{ is the convolution operation.} \quad (2)$$

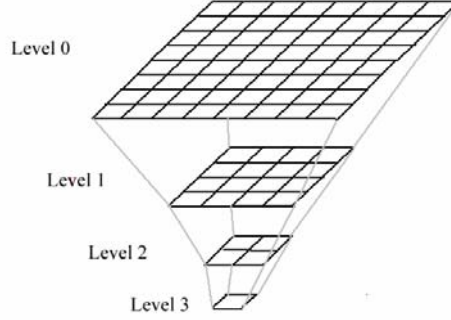
The scale parameter  $t \in \mathfrak{R}_+$  corresponds to the square of the standard deviation of the kernel  $t = \sigma^2$ . We are interested in the significant structures' edges in the image so we choose the normalised Laplacian which is a "general purpose" edge-detector. We look for maxima (with respect to  $t$ ) of  $t \nabla_x^2 L(\mathbf{x}, t)$ , where  $L$  is the scale-space representation of  $f$ , and  $f$  is the pixel intensity pattern of our image. In terms of the more usual spread of a Gaussian, we look for maxima (with respect to  $\sigma^2$ ) of  $\sigma^2 \nabla_x^2 L(x, \sigma^2)$ .

To look for these maxima, Lindeberg either: selects a fixed point (e.g., the image centre), or follows the spatial maxima through the image as they move with increasing  $t$ . To avoid the heavy processing required by the second approach while also reducing the possibility of missing scales by using the first approach, we choose several fixed points in the image. Therefore, we calculate candidate scales at 25 equally spread sample points  $\mathbf{x}_i$ . We also limit the permissible scales in the candidate scale set to between 2 and 24. Allowing higher values causes the image to be too blurred to be useful for image structure segmentation purposes.

We now have a set of candidate scales for the 25 sample points. We take the histogram of candidate scales to identify the (single) optimal scale to use to process the image and smooth this histogram with a 3-value kernel  $\{1, 2, 1\}$  to remove perturbations. The  $\{1, 2, 1\}$  kernel assigns a higher weighting to the central (chosen) value and a lower weighting to its two direct neighbours thus allowing us to select our optimum scale. The *scale* corresponding to the first highest peak in the histogram is taken as our final scale.

### 2.2 Gaussian Pyramids

In this stage the aim is to determine the informative image scales to identify structures in images. Scale-space selection identifies informative scales but can be inconsistent due to the chance placement of the 25 sample points leading to under or over generalisation of the regions surrounding each sample point. Conversely, the Gaussian Pyramid [BA83] is consistent across images but uses fixed scale values meaning it cannot adapt to different scales and may miss structures. Therefore, we introduce the pyramid as a pre-processor to provide consistency by pre-smoothing images to increase their similarity prior to scale selection.



**Figure 2. The multiple levels of the Gaussian pyramid where the filtered image levels effectively form an inverted pyramid structure.**

The pyramid takes an image  $G_0(x, y)$  and convolves the image with a Gaussian kernel (low-pass filter) to produce image  $G_1(x, y)$ . The derived image  $G_1(x, y)$  is then convolved with the kernel to produce  $G_2(x, y)$  which is then processed to produce  $G_3(x, y)$ . For our pyramid implementation, we use 4 levels  $G_0, G_1, G_2, G_3$  with dimensions 512x512, 256x256, 128x128, 64x64 pixels respectively as shown in Figure 2.

If  $I \in \mathfrak{R}^2$  is the original 512x512 pixel 2-D image then the pyramid is computed as eqs 3 and 4:

$$G_0(x, y) = I(x, y) \tag{3}$$

$$G_{i+1}(x, y) = FILTER(G_i(x, y)) + RESIZE(G_i(x, y)) \tag{4}$$

For the *FILTER* function, we use the standard Gaussian function in eq 5:

$$f_{\sigma,n}(x) = \frac{\partial^n}{\partial x^n} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}} \quad \text{where we set } \sigma^2 = 3. \tag{5}$$

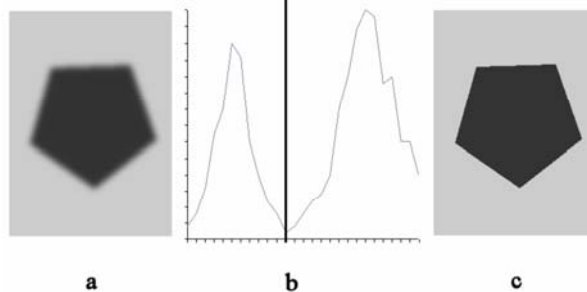
Filtering is followed by *RESIZE* which resizes  $G_i$  by scale factor 0.5 to give  $G_{i+1}$  using separable spline interpolation algorithm described in [U93]. We found that resizing without interpolation over-emphasises jagged lines in images by increasing the aliasing.

The next processing step is to divide each blurred variant of the image into regions (segmentation). We use pixel intensity categorisation to identify the structures.

### 2.3 Categorisation

To categorise (cluster) the pixels, we take our cue from Lu and Chung [LC98] who proposed a hill-clustering method for determining the number of texture clusters. So, for each pyramid level  $G_i(x, y)$ , the scale ( $\sigma$ ) is selected and the image is blurred with a Gaussian kernel of size  $\sigma$  giving  $B_i(x, y)$ . From  $B_i(x, y)$ , we generate a histogram of pixel greyscale intensity values (divided into 255 bins). This raw

histogram needs smoothing using a one-dimensional Gaussian with standard deviation 10 bins (1 pixel width) before it is usable. We then choose the  $N$  highest peaks ( $N$  categories) of the smoothed histogram and set thresholds midway between neighbouring peaks which should reflect the larger-scale structures. A simple example is shown in Figure 3.



**Figure 3. (a) is the source image. (b) is this image’s pixel intensity histogram with the pixel intensity threshold drawn for  $k=2$  categories - the trough in the histogram identifies the threshold (category boundary). (c) shows the result of categorisation.**

Previous pixel categorisation work [MJ92] tends to rely upon a pre-specified maximum number of categories  $M_{max}$ . The optimum number of categories is then determined by segmenting the image into  $k$  categories for  $2 \leq k \leq M_{max}$  and using some suitable criterion to select the optimum [MJ92] which is laborious. We employ a simple heuristic which we developed following detailed analysis of the pixel intensity peaks of 450 trademark images used in [LDH07]: sort the peaks into peak intensity order and if the peak value is less than 100 then do not include the peak. This resets  $M_{max}$  to the  $k$  peaks with values greater than 100. This value (100) was derived through a series of analyses. It is a trade-off: too high a value causes some images to have too few or even 0 categorisations. Too low a value causes too many categorisations for some images. We then identify the 2 highest peaks, 3 highest peaks up to  $M_{max}$  highest peaks and divide the image into a series of views (image representations) with 2, 3 ...  $M_{max}$  categories per view. The result is a series of categorised views where pixels of similar intensity are grouped to reveal the structures within the image.

## 2.4 View Generation

This part of our approach differentiates different types of images. In particular images containing texture from those containing no textures (just lines and block filled regions). This is done because later processing requires different processes applied to the two types of images. In particular, line and region images require merging of lower level image structures (shapes) to infer the higher level structures where as textured and noisy images require the texture or noise to be effectively blurred out to produce a homogeneous region to represent the structure (shapes and regions) in the image.

Using the methods in 2.3 we specify  $M_{max}$  as 2 for line and region-based images that are bicolour (black and white) and  $M_{max}$  as 4 for texture/noisy or grey-scale images. Note  $M_{max}$  may be reset if there are fewer than 4 peaks over 100. We have erred on the side of caution by allowing 4 categories to ensure all views are found while potentially some unwanted views may be generated.

For the classification process we first apply a Laplacian pyramid  $L_0$  operator, which represents the difference of Gaussians ( $G_0 - G_1$ ) [BA83]. This is essentially an edge detection of  $G_0$  and is given in eq 6:

$$L_0(x, y) = G_0(x, y) - RESIZE(G_1(x, y)) \quad (6)$$

Using the  $L_0$  we can differentiate the types of images as textured images will have a higher energy (more edges) compared to line/region images. To set the threshold on  $L_0$  we used visual analyses of the energy levels of the decompositions seen by humans in 84 trademark images in a set of experiments [HEA06, HHEA06], the decompositions seen by humans in 63 trademark images in a set of experiments [REB00] and a further set of 450 images comprising clean, noisy and textured images [LDH07].

Over-all we use the following processing steps for the two types of images:

First, calculate the energy of  $L_0$  as in eq. 7.

$$Energy = \sqrt{\sum_{\forall x,y} p(x,y)^2} \quad \text{where } p(x,y) \text{ is the greyscale value of pixel } (x,y) \text{ in } L_0. \quad (7)$$

Then apply the following decision rules:

If  $energy < 9600$  then process the image as a region-based/line-based.

If  $energy \geq 9600$  then process the image as a textured/noisy image.

Following identification of the two types of images we then apply two different processes:

#### 2.4.1 For region/line-based images

- $G_0$  – unprocessed.
- $G_2$  – straight categorisation of  $G_2$  image – no scale selection.
- $G_3$  – select scale (kernel width), convolve Gaussian ( $\sigma$ ) with  $G_3$  image, categorise resulting convolved image.

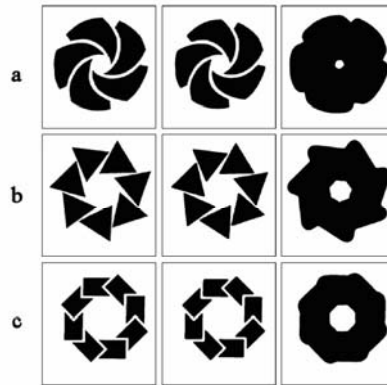
#### 2.4.2 For texture/noisy images

There is a tendency for  $\sigma_0 = \sigma_2$  in textured/noisy images where  $\sigma_0$  is the scale selected for  $G_0$  and  $\sigma_2$  is the scale selected for  $G_2$ . During our analyses, we found that  $G_0$  and  $G_2$  were the best levels of the Gaussian pyramid to process for textured images. However, if  $\sigma_0 = \sigma_2$  this would produce virtually identical outputs when  $G_0$  and  $G_2$  were convolved with equivalent kernels and is not desirable. Accordingly, we test for equivalence and alter our processing strategy accordingly.

- If ( $\sigma_0 < \sigma_2$ ) then
  - $G_0$  – select scale (kernel width), convolve Gaussian ( $\sigma_0$ ) with  $G_0$  image, categorise resulting convolved image.
  - $G_2$  – select scale (kernel width), convolve Gaussian ( $\sigma_2$ ) with  $G_2$  image, categorise resulting convolved image.
- If ( $\sigma_0 = \sigma_2$ ) then
  - $G_0$  – select scale (kernel width), convolve Gaussian ( $\sigma_0$ ) with  $G_0$  image, categorise resulting convolved image.
  - $G_3$  – straight categorisation of  $G_3$  – no scale selection.

## 2.5 Results for view generation

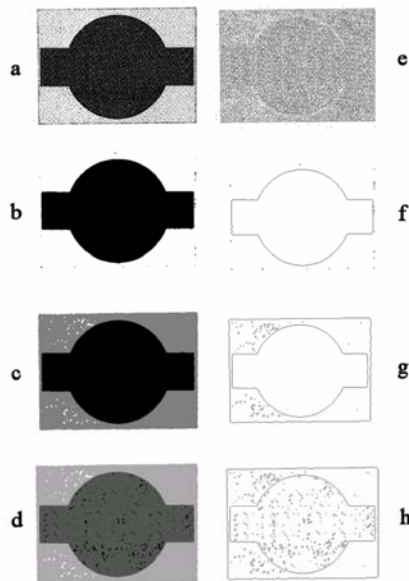
We present some results of our view generation processing pipeline to demonstrate that the desired structures are being found and that texture and noise are effectively merged to form definitive shapes. Figure 4 shows that higher-level structure (a ring shape) is extracted using blurring and categorisation. In Figure 5, we show the result of blurring and categorising a textured and noisy image to demonstrate that the texture is clustered and the higher-level structure of the image is revealed.



**Figure 4. Three images (a, b and c) and their respective outputs. All images were classified as line/region by the energy-based classifier.**

In Figure 4, the views produced from each image are similar when compared visually by a human observer on a column basis. The ring-structure has been found. If the three images in Figure 4, column 3 were matched the ring structures would be similar. If the three original images in column 1 were matched they would not be similar.

Our results are not perfect. For example, in Figure 5, results (b) and (c) are good. View (d) is probably superfluous here but the energy level and pixel intensity minimum have to be set globally so this may result in an occasional superfluous output for some images. The edges shown in (f,) (g) and (h) demonstrate that we have found the image structures to allow image matching. Although there is a tiny amount of noise remaining, it comprises very small blobs which could easily be removed using a suitable image processing technique. In contrast, image (e) shows the (1000+) edges detected in the original image and no discernible structures.



**Figure 5.** The original image (a) is processed to produce a series of image views (b, c and d). The edges found are shown in e, f, g, and h

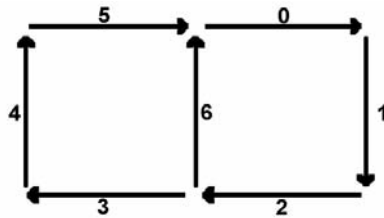
## 2.6 Shape Identification

In sections 2.1-2.5, we have produced various views of an image with the aim of merging lower level shapes and texture to pinpoint perceptual structures. Next we identify shapes in this data. Our image structure-finding approach uses a closed shape identification algorithm. The method adapts and refines Saund’s closed shape identification algorithm [S03]. By doing this, the approach can find higher level (perceptual) shapes.

Initially, the closed shape algorithm requires an underlying technique to identify the edge segments within an image and to detect the relationships between those edge segments. We resize the multiple views generated to 2048x2048 pixels from 512x512 to ensure edge separation as all structures will be at least 4 pixels wide and the structure’s edges will not be adjacent. If the edges are in adjacent pixels then tracing the shapes is difficult as it is not clear which edge a pixel belongs to. We resize with no interpolation to prevent blurring of the edges in the view as blurred edges will confuse the edge detector. We find the edges in the image using a simple Laplacian edge detector before subdividing these edges into constant curvature segments (CCSs) using the Wuescher & Boyer [WB91] curve segmentation algorithm. This aggregates edge primitives into more perceptually-oriented CCSs. We have refined and improved the technique by increasing the tidying of the edges prior to edge segmentation to ensure there are no gaps or errors in the edges and tailoring the parameter settings for trademark images to improve the quality of the CCSs produced. We set the scale parameter to 2.4 following extensive evaluation of the CCSs generated for the images used in [HEA06, HHEA06] at different scale parameter settings. If the image view resolves to 600 CCSs or more then we do not process the image view. This is effectively a safeguard as processing high numbers of CCSs is very slow and image views that contain 600 CCSs or more generally contain textured regions that have not been blurred sufficiently.

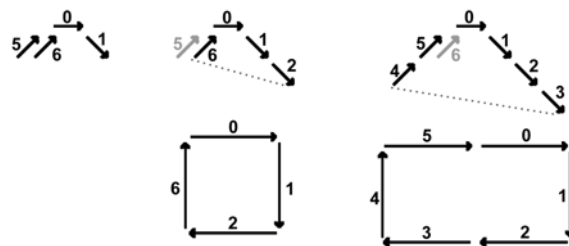
If there are fewer than 600 CCSs then the CCSs are used in two ways. First, they are used in the lower level graph representation in our image matching phase described in section 3 where the CCSs form the nodes in the graph and the relationships between the CCSs form the arcs in the graph.

Secondly, the CCSs are also the building blocks for our closed shape identifier as illustrated in Figure 6. Our aim is to group these CCSs using Gestalt-like methods to produce a graph of CCS relations which will underpin the closed shape identification algorithm. Each CCS becomes a node in the graph. Each CCS has two labelled **end** points (**first** point - denoted as an x, y coordinate and **last** point - also denoted as an x, y coordinate). In the following: **end** denotes an end point (either first or last), **first** denotes the end point (x, y coordinate) labelled first in a particular CCS and **last** denotes the end point labelled last. We find all CCSs that are end-point proximal. We extract endpoint proximity by comparing the distance between end points of CCSs, that is the distances between all points in the set of all {**first** and **last**}. We have evaluated various distances (in pixels) to use for end-point proximity calculations and found the following performed optimally with respect to finding perceptual shapes and structures. We favour collinear proximity. Two collinear CCSs are proximal if they have a large gap but we reduce this gap if they are not collinear. Humans favour larger gaps in collinear lines [KS91].



**Figure 6. A set of CCSs (0-6). The arrow heads denote the first end of the line segment and the opposite end of the line segment is hence the last end.**













Our closed shape algorithm overlays this graph. The search commences from each end (first and last) of each node (CCS). For each end (first then last) in turn, all possible paths are followed. This effectively forms a search tree with paths through the tree representing the possible shapes present in the image, see Figure 6.



**Figure 7. The search tree for the set of CCSs in Figure 6. The left tree shows the tree after expanding each end of node 0 (root). The middle tree shows how, when the tree is expanded by node 2, a closed path is found - 0126. When 2 is expanded, although 6 is end-point proximal it is not added as it is already present on the opposite side of the tree. The right tree shows the tree expanded by node 4 and node 3. A second closed path is identified - 012345.**

The search is managed through the use of scores for ranking possible paths through junctions such as t-junction or crossroads, see table 1. We have revised the junction scores used by Saund to improve the quality of the results for figurative images and to make the algorithm more consistent. We used the results from our previous work involving human experiments [HEA06, HHEA06] to derive our new

junction scores. During path search and scoring, we separate straight paths from turning paths using the table of scores depending on whether the path is: turning clockwise (CW) or anticlockwise (ACW); OR straight clockwise or anticlockwise. Each path accumulates a score using the score from each junction it passes through. Our path scores are an average of the junction scores. Saund's uses a cumulative (product) calculation but this favours short paths whereas we allow longer paths to be explored. We have a minimum score threshold (ScoreForStraightPaths = 0.6 and ScoreForTurningPaths = 0.8), compared to 0.6 and 0.9 respectively for Saund. As soon as the average score for a path falls below the minimum score, we terminate the search on that path. These minimum scores were derived from a series of analyses using the images from [LDH07]. Terminating the search as soon as the average score falls below the threshold value allows us to speed the graph search by reducing the graph search space and prevents poor quality paths being followed.

Junction	Turning ACW	Turning CW	Straight ACW	Straight CW
$dist(CCS_1, CCS_2) < 2 \text{ pixels}$	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>
	1.0	0.7	1.0	0.7
	0.7	1.0	0.7	1.0
	1.0	0.5	0.9	0.5
	<b>1.0</b>	1.0	1.0	1.0
	1.0	<b>1.0</b>	1.0	1.0
	0.5	1.0	0.5	0.9
	1.0	0.5	0.9	0.5
	0.5	1.0	0.5	0.9
	1.0	0.5	0.6	0.5
	<b>1.0</b>	<b>1.0</b>	1.0	1.0
	0.5	1.0	0.5	0.6
	1.0	1.0	1.0	1.0

**Table 1. A table of the shape finding junction scores. Each row represents a junction configuration such as t-junction or crossroads. The arrow indicates the path direction through the junction. The bold scores differ from Saund's scores.**

As each leaf node in the tree is expanded, new child nodes are compared with child nodes in the opposite side of the tree. If they are end-point proximal then a closed path (a shape) has been identified and its nodes and boundary pixels are added to the list of candidate paths.

There are potentially multiple iterations of the shape finding process to ensure that not too many shapes are output. On each iteration the the gap jumping parameters are reduced along with the shape finding parameter settings on each sub-iteration, as follows:



- Iteration 1: on the first iteration, we set the permissible gaps at 32 pixels and 256 pixels. If  $\text{dist}(\text{CCS}_{1\_end}, \text{CCS}_{2\_end}) < 32$  pixels then  $\text{CCS}_{1\_end}$  and  $\text{CCS}_{2\_end}$  are end-point proximal. If  $\text{dist}(\text{CCS}_{1\_end}, \text{CCS}_{2\_end}) < 256$  and the difference between the gradients of the lines (or the terminal gradients of curves) is within  $\pm 5^\circ$  then  $\text{CCS}_{1\_end}$  and  $\text{CCS}_{2\_end}$  are end-point proximal (and continuous). This effectively joins the graph by linking the proximal end-points and mimics human perception by allowing a wider gap between continuous pairs than non-continuous pairs of CCSs. Also, we differentiate CCS ends (first, last) and only allow one end-point proximity between  $\text{CCS}_{1\_last}$  and an end-point of  $\text{CCS}_2$  and one end-point proximity between  $\text{CCS}_{1\_first}$  and an end-point of  $\text{CCS}_2$  to prevent cycles and vice versa. We always use the closest so if  $\text{dist}(\text{CCS}_{1\_last}, \text{CCS}_{2\_first})=10$ ,  $\text{dist}(\text{CCS}_{1\_last}, \text{CCS}_{2\_last})=11$ ,  $\text{dist}(\text{CCS}_{1\_first}, \text{CCS}_{2\_first})=11$ ,  $\text{dist}(\text{CCS}_{1\_first}, \text{CCS}_{2\_last})=12$  then the proximity is  $\text{CCS}_{1\_last} \rightarrow \text{CCS}_{2\_first}$  and  $\text{CCS}_{1\_first} \rightarrow \text{CCS}_{2\_last}$  even though  $\text{dist}(\text{CCS}_{1\_last}, \text{CCS}_{2\_last}) < 32$  and  $\text{dist}(\text{CCS}_{1\_first}, \text{CCS}_{2\_first}) < 32$ .
  - If there are fewer than 200 shapes found after shape finding  $\{\text{CandidateShapeSet}_y\}$  for  $\text{imageView}_y$  then we run the similarity pruning step described later in section 2.6.1.
    - If there are fewer than 100 shapes after similarity pruning then the shapes are output  $\{\text{ShapeSet}_y\}$  and processing terminates.
  - If there are over 200 shapes found by the shape finding before pruning or more than 100 shapes following pruning then we revise the parameter settings upwards and rerun the shape finder ( $\text{ScoreForStraightPaths} += 0.1$  and  $\text{ScoreForTurningPaths} += 0.1$ ),. We continue revising the parameter settings upward until either fewer than 200 shapes are found or the parameter setting reach their maximum permitted values ( $\text{MAX\_ScoreForStraightPaths} = 0.8$  and  $\text{MAX\_ScoreForTurningPaths} = 1.0$ ). This MAX score guides the iterations.
    - If parameters have reached their maxima then run iteration 2.
  - If there are fewer than 200 shapes found after revising the parameters for shape finding to produce  $\{\text{CandidateShapeSet}_y\}$  for  $\text{imageView}_y$  then we run the similarity pruning step described later in section 2.6.1.
    - If there are fewer than 100 shapes after similarity pruning then the shapes are output  $\{\text{ShapeSet}_y\}$  and processing terminates.
    - If there are 100 shapes or more then run iteration 2.
- Iteration 2: we decrease the gaps permitted between the CCSs. A high number of shapes are undesirable for image matching as it increases the data size and the search space for comparing images on a shape matching basis and thus causes the matching algorithms to run too slowly. By decreasing the gaps, the shape finder does not find the perceptual shapes but becomes a shape tracer that finds the basic shapes present in the image. There will generally be fewer basic shapes than perceptual shapes so the search space will be decreased. The new gaps are 8 pixels for continuous and proximal CCSs and 4 pixels for proximal but not continuous CCSs and  $\{\text{CandidateShapeSet}_y\}$  are produced.
  - If there are fewer than 200 shapes found after shape finding  $\{\text{CandidateShapeSet}_y\}$  for  $\text{imageView}_y$  then we run the similarity pruning step described later in section 2.6.1.
    - If there are fewer than 100 shapes after similarity pruning then the shapes are output  $\{\text{ShapeSet}_y\}$  and processing terminates.
  - If there are over 200 shapes found by the shape finding before pruning or more than 100 shapes following pruning then we revise the parameter settings upwards and rerun the shape finder ( $\text{ScoreForStraightPaths} += 0.1$  and  $\text{ScoreForTurningPaths} += 0.1$ ),. We continue revising the parameter settings upward until either fewer than 200 shapes are found or the parameter setting reach their maximum permitted values ( $\text{MAX\_ScoreForStraightPaths} = 0.8$  and  $\text{MAX\_ScoreForTurningPaths} = 1.0$ ).

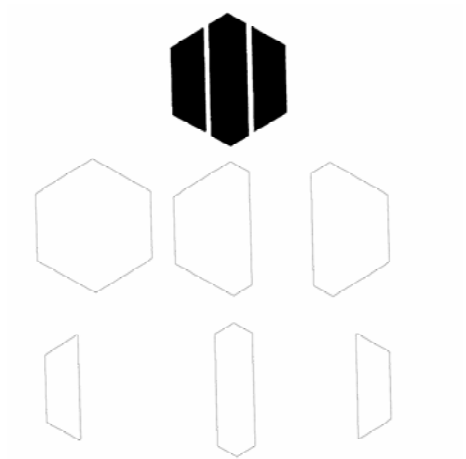
- If there are fewer than 200 shapes found after revising the parameters for shape finding to produce  $\{\text{CandidateShapeSet}_y\}$  for  $\text{imageView}_y$  then we run the similarity pruning step described later in section 2.6.1.
  - If there are fewer than 100 shapes after similarity pruning then the shapes are output  $\{\text{ShapeSet}_y\}$  and processing terminates.
  - If there are 100 shapes or more then no output is produced and processing terminates.

### 2.6.1 Similarity Pruning

The set of candidate paths generated above forms the initial set of shapes  $\{\text{CandidateShapeSet}_y\}$  for the image view ( $\text{imageView}_y$ ). For iteration 1 with large gaps between the CCSs (32 and 256 pixels), any shapes with an area of fewer than 2500 pixels are discarded as they are too small to be noticeable in a 2048x2048 pixel image view. For iteration 2 with small gaps between the CCSs (4 and 8 pixels), any shapes with an area of fewer than 900 pixels are discarded as they are too small to be noticeable in a 2048x2048 pixel image view. We then run similarity pruning to produce the final set of shapes for each image. We have identified a set of features to determine when shapes output by the closed shape identification algorithm are very similar and hence when one shape should be subsumed by the other. If the top, bottom, left and right coordinates of the axis-aligned bounding box are within 1 pixel, the lengths of the perimeters of the two shapes are within 10 pixels and the area ratio for the two shapes is over 0.95 then we keep the shape with the highest perceptual score. We use the perceptual shape classifier developed in [HEA\_CIVR07] to generate a perceptual score  $0 \leq \text{score} \leq 1$  for each shape. We do not use the shape classifier to classify shapes here, only to score them. All shapes left after post-processing represent the generated shape set  $\{\text{ShapeSet}_y\}$  for that image view ( $\text{imageView}_y$ ). If this produces fewer than 100 shapes then these are output. If there are more than 100 shapes then there are no outputs for the image view as the particularly image view has not been blurred sufficiently or contains too much noise to be processed correctly.

### 2.7 Results for perceptual shape finding

The following presents some results of our shape finding methods to demonstrate that they are finding the desired shapes in the images output from the blurring and categorisation stages. In Figure 8, we show that perceptual shapes from an image output by our processing pipeline are found using our methods. We thus demonstrate that by using our processing pathway to blur, categorise, edge segment and identify the shapes, perceptually relevant shapes may be extracted.

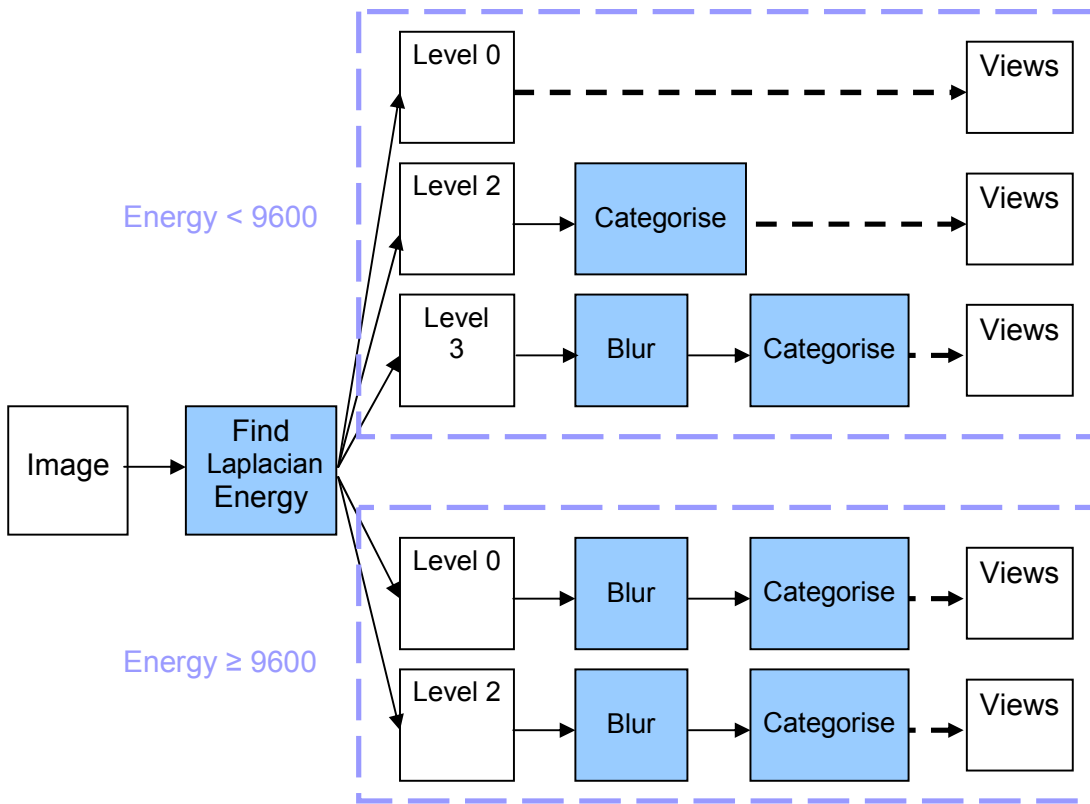


**Figure 8. The six perceptual shapes found by the shape identifier from the trademark image view in the top row.**

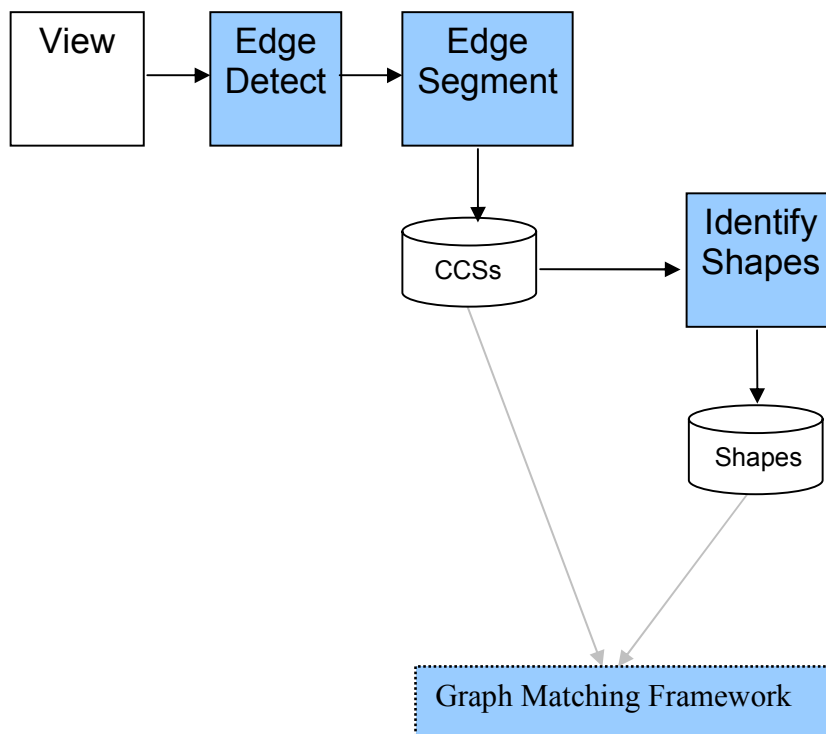
In Figure 8, the shape identifier has found the set of perceptual shapes we may expect a human to identify [HHEA06, HEA06] in the trademark image view. This set of shapes may be used for perceptual image matching and retrieval and will form part of the input data for the graph matching framework described next.

### 3 The Graph Matching Framework

The graph matching framework processes the query image to produce views at various perceptual levels (see Figure 9), finds the shapes and structures in those views (see Figure 10) and measures the similarity between the query image’s data and the data of the stored images. Finally, the system ranks the matching stored images by their similarity and returns the results to the user. Our graph matching framework extracts the data to represent each image in the database *offline* and then performs *online* query processing. By constructing the database offline, we ensure maximum efficiency as the database is only processed once during this initial database pre-processing phase. The data produced for each image may be stored with the image in the database and then queried online. When a query image is presented to the system during query processing, the stored data for each database image is immediately available for matching so query processing is maximally efficient. A user may query the system to find the best matching images by inputting a query image.



**Figure 9.** This phase of the image processing pipeline initially identifies the likely image type – whether the image may be considered a line/region image or a noisy/textured image. We use the image’s energy to differentiate the two types so images with energy of less than 9600 are treated as line/region and images with energy equal to or greater than 9600 being treated as noisy/textured. The two types are then processed differently. More blurring is applied to the noisy/textured images to smooth the texture or noise and allow the image structures to be found. The original image is not produced as an output view for textured/noisy images otherwise the edge detector which we use to process the image views would find large numbers of edges within the texture or noise which is not desirable. The line/region images are blurred less and the original image is output as a view.

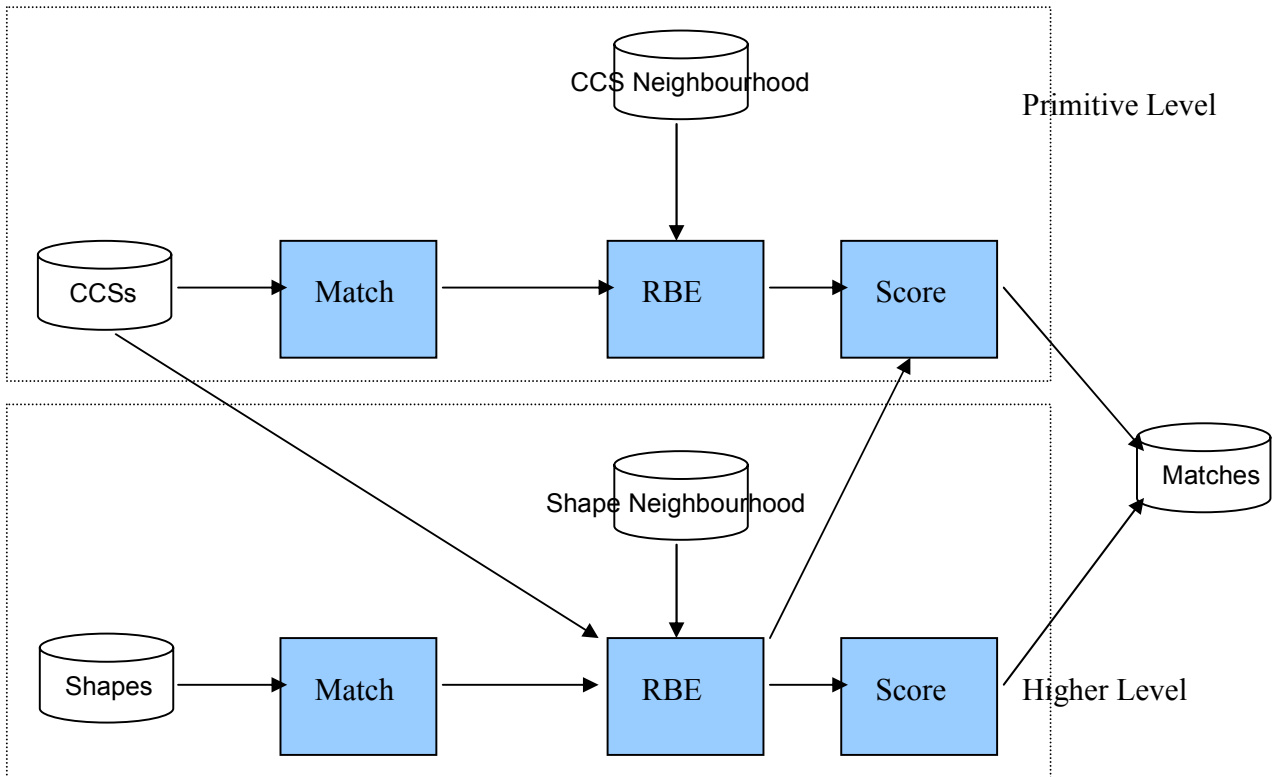


**Figure 10.** Once the image has been converted to a number of views (representations) then each view is processed to find the shapes present. This shape finding uses edge detect, segment and shape find. There are two output files produced for the PROFIT project: a file listing the constant curvature segments (CCSs) and a file detailing the shapes found (see Appendix for examples). The CCSs are edge segments – the edge detector finds the edges in the image and splits them into constant curvature segments. The shapes are formed by joining CCSs to form closed figures.

As stated in the introduction, we have adopted a geometric image matching and retrieval approach that uses the CCSs and perceptual shapes/structures identified in the previous phase to form a bi-level representation of each image, which are then represented as a graph. The graph matching method chosen is based on approach developed by Alwis [A99].

The matching process commences by applying our processing pipeline in Fig 9 and 10. This produces multiple representations (multiple image outputs as shown in Figure 4 and 5). Each of these image outputs is stored in the database as they represent the images at varying levels and allow us to perform multi-level matching. [E00] identified this multilevel approach as a key future development of trademark image retrieval. Biederman et al. [BSBKF99] propose that human image recognition works on various levels. Their analyses suggest that humans decompose images into components. These components have qualitative ('non-accidental') properties and relations that allow images to be matched. The authors determine experimentally that qualitative properties have more influence on object matching (whether two images are deemed similar) than quantitative. The authors [BSBKF99] go on to posit a hierarchical architecture for image decomposition building up from image primitives that the authors termed 'geons'. Following Biederman's proposal, there are then two levels of matching employed in our image matching and retrieval system. For the image matching, we use the CCSs and

shapes/structures from the various image representations as our data and use evidence combination to produce an overall score for each image. We use perceptual relationships between the CCSs as our lower level matching and features calculated from the shapes as a higher level matching.



**Figure 11. The CCSs are edge segments – the edge detector finds the edges in the image and splits them into constant curvature segments and form the primitive level of matching in the graph matcher. The shapes are formed by joining CCSs to form closed figures and these are used by the graph matcher in the higher level matching.**

To perform matching of the multi-level representations of the trademark images generated by our view generation phase using the primitives (CCSs) as the data, we use a graph matching process using the Relaxation by Elimination algorithm (RBE) in conjunction with evidence counting [A99]. We also use a similar technique to match views at the shape level and incorporate links to the CCSs that form each shape to percolate evidence up to the higher (shape) level from the primitive (CCS) level. RBE and the graph matcher provide an efficient matching framework for image matching using a multi-level retrieval mechanism [A99].

The initial stage of this process is to generate a graph from the primitives to represent an image. In section 2.6 we detailed how image edges are segmented into CCSs. Each CCS has the following attributes associated:

- CCS identifier
- Type - curve/line
- Length - straight-line distance between end-points in pixels
- Orientation - gradient of line (not set for curves)
- Curvature -  $\frac{1}{\text{radius}}$  (not set for lines)
- Perimeter - length of arc formed by curve (not set for lines)

In our shape finding algorithm, we calculated relationships between the CCSs which we use here as the primitives. The relationships calculated between CCSs were:

- Collinearity – the CCSs have gradients (or terminal gradients for curves) that are within  $\pm 5^\circ$ .
- Proximity – the end points of the CCSs are within 32 pixels or 256 pixels if the CCSs are collinear.
- We augment this information with corners which are defined as two proximal CCSs that are not collinear.

These relationships thus mimic the proposed Gestalt principles of proximity and continuity thus allowing our system to mimic the Gestalt system for human image matching. We can now produce a graph to represent each image. The CCSs are the nodes and the arcs are the three relationships, collinear, proximity, and corner. Two nodes are joined by a collinear arc if their gradients are within  $\pm 5^\circ$ . They are joined by a proximal arc if their end-points are within 32 pixels or 256 pixels if they are collinear. They are joined by a corner arc if they are proximal but not collinear.

Our graph matching framework uses descriptions [A99] of the image in which the graph nodes represent the primitives (CCSs of the image) and the graph arcs describe different perceptual relationships between those CCSs.

The RBE based matching process aims to match the query graph to a large set of previously stored graphs representing the trademark images.

The first stage of RBE generates an initial set of candidate matches by comparing nodes of the query graph against the various nodes of the model graphs stored in the database using local features of the CCSs (type, length, orientation, curvature and perimeter) and also links to shapes that are built using the particular CCS. Feature vector comparison can be performed in two different ways; either in a symbolic fashion by discretising the feature vector elements or by calculating distance measures between the corresponding feature vector elements of the query image and the database image. Here, we discretise the vector elements and match the feature vector values using the discretised bins.

Once an initial set of candidate matches is obtained RBE accumulates evidence from each of the nodes connected neighbours (representing CCS's perceptual neighbourhood) and iteratively eliminates implausible candidate matches using upper bound probability estimations. During this process, the query graph is compared against all of the image representation graphs in the database and each stored graph accumulates a match score through the use of evidence counting. To minimize the search space to maintain computational tractability, we only use evidence from each node's perceptual neighbours rather than counting evidence from all of the neighbours in the graph structure as performed in the standard RBE process [TA98]. The neighbourhood comprises other CCSs connected by the three relationships, collinear, proximity, and corner and also links through CCS→shape relations.

The support value for solution  $\alpha$  at CCS  $i$  after iteration  $n$  is given by:

$$S^n(\theta_i=\alpha) = \sum_{N_i} \max_{\beta \in \Theta_j^n} h(\theta_i=\alpha, \theta_j=\beta)$$

Where  $N_i$  is the perceptual neighbourhood for CCS  $i$ ,  $\Theta_j^n$  is the set of remaining candidates for CCS  $j$  at iteration  $n$  and  $h(\theta_i=\alpha, \theta_j=\beta)$  is a binary compatibility measure between solutions  $\theta_i=\alpha$  and  $\theta_j=\beta$ .

A solution is eliminated if its support value is below a threshold  $\lambda$ .

Eliminate if  $S^n(\theta_i=\alpha) \leq \lambda$

RBE is an iterative process which stops when the process obtains stability (i.e. no more implausible candidate eliminations). The remaining candidates that have not been eliminated are then used to calculate a final similarity score for the primitive level (CCS level) and the images are ranked; sorted by score.

In the previous paragraph, we described our image matching mechanism using the image primitives as the data. We perform two levels of image matching within our graph matching framework to continue our plan of using multi-level image representations at all stages of the matching process to mimic the multi-level matching employed by humans [E00]. The second matching level is based on closed shapes/structures, we use feature vectors to represent each shape/structure and incorporate information from the other shapes/structures identified within the image along with connections to the CCS used to construct the various shapes and structures. Shapes are matched against shapes using partial matching of feature vectors which is much less affected by the number of shapes/structures found by our shape detection algorithm within the images than other shape matching techniques. The inability to accommodate images with differing numbers of shapes is a drawback we observed in global image matching methods such as deformable image templates [DP97].

We represent each closed shape/structure using the following features:

- Shape identifier
- Circularity where  $\text{Circularity} = 4\pi * \text{Area} / \text{Perimeter}$
- Convex Hull Perimeter
- Roughness where  $\text{Roughness} = \text{Perimeter} / \text{Convex Hull Perimeter}$
- We also store the angle between each shape's centroid and the centroid of every other shape in the image which produces a shape angle matrix.
- Finally, we store the CCS identifiers of the CCSs that form each shape found by our shape identification algorithm.

We calculate the similarity between a query image and the stored images by using the CCS→shape links and the feature vectors of each shape/structure to produce an initial set of candidate matches for each shape/structure in the query image. As before, feature vector comparison can be performed in two different ways; either in a symbolic fashion by discretising the feature vector elements or by calculating distance measures between the corresponding feature vector elements of the query image and the database image. Here, we discretise the vector elements and then using matching bin counts to accumulate support. The graph matcher also uses CCS→shape relations to accumulate support. We then iteratively accumulate support for each matching possibility from the shape's contextual neighbourhood (linked shapes and CCSs) and unlikely candidates are eliminated using upper bound probability estimations from the query node's perceptual neighbours. This is an iterative process which is stopped when the process obtains stability (i.e. no more elimination).

The support value for solution  $\alpha$  at shape  $i$  after iteration  $n$  is given by:

$$S^n(\theta_i=\alpha) = \sum_{N_i} \max_{\beta \in \Theta_j^n} h(\theta_i=\alpha, \theta_j=\beta)$$

Where  $N_i$  is the perceptual neighbourhood for shape  $i$ ,  $\Theta_j^n$  is the set of remaining candidates for shape  $j$  at iteration  $n$  and  $h(\theta_i=\alpha, \theta_j=\beta)$  is a binary compatibility measure between solutions  $\theta_i=\alpha$  and  $\theta_j=\beta$ .



A solution is eliminated if its support value is below a threshold  $\lambda$ .

Eliminate if  $S^n(\theta_i=\alpha) \leq \lambda$

The remaining candidates are then used to calculate a final similarity score for the higher (shape) level using evidence obtained from both the feature vectors and the shape's neighbourhood for each image.

We aim to produce a unified multi-level image retrieval system using multiple image representations output by the view generation pipeline and multilevel representations within the actual image matching. We require an efficient and accurate method to combine the results from the multiple levels in the graph representation. Image retrieval results produced using different representations show that no single representation excels but that a combination of representations would be preferable [E00]. The graph matching framework applies a 'results level' approach using rankings from the two matching levels. Alwis and Austin [AA99] posited that this approach produced better results than alternative representations and had been widely adopted in the document retrieval field where evidence from different document matching levels need to be combined. Alwis [A99] assigns more weight to the top matches in the list and combines the multiple levels using reciprocals of ranks. The combined match score is calculated as follows:

$$\text{combined score} = \sum_i \frac{1}{\text{rank}_i}$$

Where  $\text{rank}_i$  is the image's rank calculated using the graph matching retrieval level  $i$ .

Alwis [Thesis] proposed a framework for combining similarity scores using the Dempster-Shafer method. The Dempster-Shafer method combines similarity scores from two matching levels using the following equation:

$$\text{CS}_k = \mu_1(s_k^1/S_1) \mu_2(s_k^2/S_2) + (1 - \mu_2) \mu_1(s_k^1/S_1) + (1 - \mu_1) \mu_2(s_k^2/S_2)$$

Where  $\text{CS}_k$  represents the combined similarity score (combined from two levels) for image  $k$

- $\mu_i$  is the belief from level  $i$
- $s_k^i$  represents the similarity score for image  $k$  at level  $i$
- and  $S_i = \sum_k s_k^i$

## 4 Evaluation

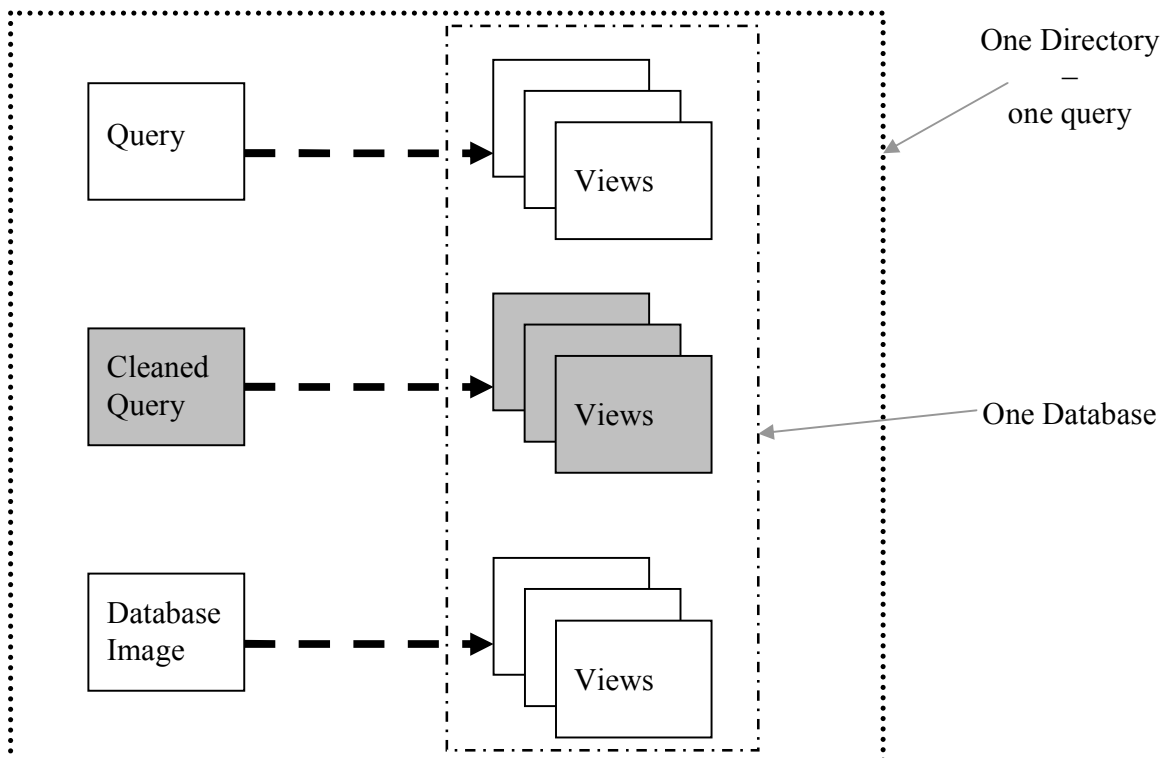
We analyse the graph matcher to investigate the recall and precision when retrieving matches for query images.

### 4.1 Data

For this evaluation we use the Aktor data set. The Aktor final distribution of the ground truth was supplied 7<sup>th</sup> April, 2007 as a zip file (device.zip) which the partners could download from the Aktor website. This data comprises 262 query images (order.tiff) in 262 separate directories. There are 50497 (TIFF) image files in total in the data set. In each directory are a query image (order.tiff), a number of retrieved images which were retrieved using their Vienna classification and a text file (marksList.txt). In some directories there is cleaned version of the query image with noise removed order\_cleaned.jpg. The text file (marksList.txt) lists which images within that directory are relevant for that query image according to human judgment and which images within that directory are not relevant for that query according to human judgment.

For the evaluation here we selected 34 directories which were recommended as most relevant by Aktor. These 34 directories hold 34 query images and 3164 retrieved (database) images giving 3198 images. There are also 27 cleaned query images (order\_cleaned.jpg) where noise and texture have been removed from the original query image which results in 3225 images in total.

We blurred these 3225 images (both query, cleaned-query and database images) to produce 11501 image views across the 34 directories. This produced blurred query views, blurred cleaned-query views and blurred database image views as shown in *Figure 2*.



**Figure 12.** Figure showing the view generation process for one directory (of the 34 directories). The generated views are used to form the database for that query. Not all directories have a cleaned query so this is shown in grey shading.

## 4.2 Evaluation Framework

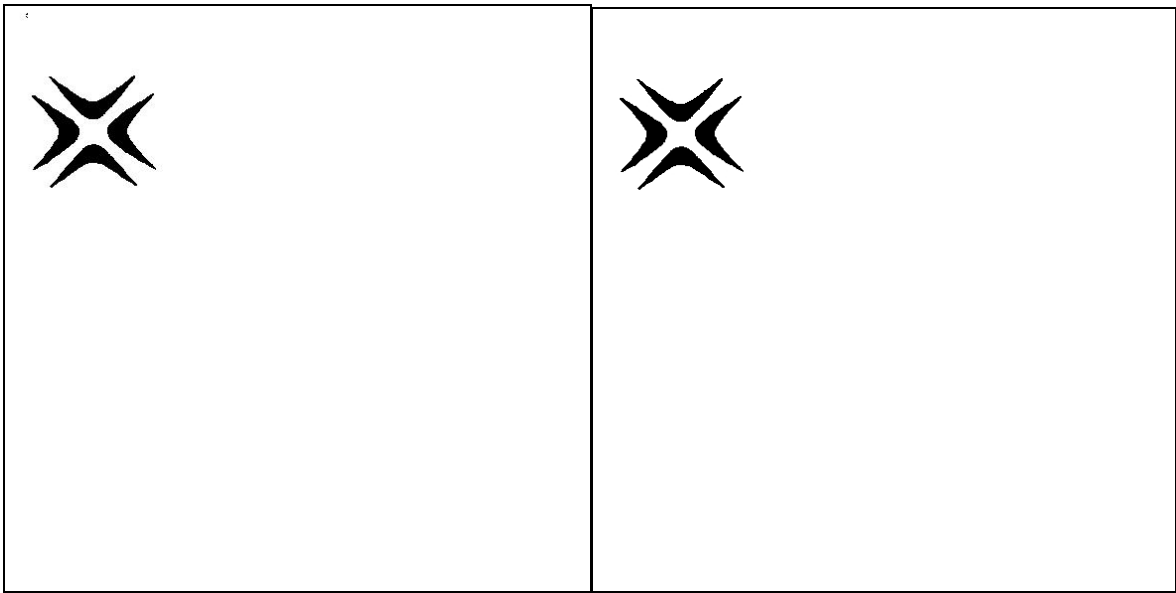
Each directory contains a text file (supplied by Aktor) listing which database images from that directory are judged relevant (in scope) for the query and which are judged irrelevant (not in scope). Thus the set of all images in the directory may act as a database and be trained and indexed by the graph matcher. We may then apply the query image from that directory against the stored images and rank the images using the graph matcher scores for each image. These image ranks may be compared to the text file listing the expected best matches (in scope images) to produce recall and precision figures for that query. There are 34 directors in total so this will produce 34 recall and 34 precision figures.

If there is a cleaned query image available (supplied by Aktor) then we use this as our query image for that directory (database) otherwise we use the standard query image.

Our evaluation framework is complicated by the fact that we are blurring images to produce multiple views (as shown in figure 12) so this multi-representation has to be accommodated in the evaluation framework. To accommodate multiple views in the query image, we apply each view to the database and retrieve a set of matches for each query view. For each set of matches, we take the highest score for each database image in that set of matches. For example, if there are three query views (qv1, qv2, qv3) and database image 1 (db1) has 2 views (db1\_1, db1\_2) and database image 2 (db2) has 2 views (db2\_1, db2\_2). We apply qv1 and return scores for all image views (db1\_1=0.5, db1\_2=**0.6**, db2\_1=**0.9**, db2\_2=0.3). We apply qv2 and return scores for all image views (db1\_1=**0.9**, db1\_2=0.4, db2\_1=0.5, db2\_2=**0.6**). We apply qv3 and return scores for all image views (db1\_1=0.5, db1\_2=**0.7**, db2\_1=0.2, db2\_2=**0.3**). Thus the highest score for db1 for qv1 is 0.6, for qv2 is 0.9 and for qv3 is 0.7. The highest score for db2 for qv1 is 0.9, for qv2 is 0.6 and for qv3 is 0.3. The cumulative total score for each database image is then the sum of scores for that image across all sets of matches. The cumulative total score for db1 is  $0.6+0.9+0.7=2.2$ . The cumulative total score for db2 is  $0.9+0.6+0.3=1.8$ . The database images may then be ranked from 1... N by sorting them on the cumulative score.

There were some problems with the data which affected our evaluation so we note these here.

1. In directory 23300, the query image had to be hand edited by York as the supplied cleaned query image would not produce any outputs (image views) from our blurring process. The supplied image is shown in Figure 13 only contains a very small structure in the top left corner and this structure is very noisy (there is a large amount of light grey noise surrounding the black regions). We removed the noise and were able to generate an image view.



**Figure 13.** The figure shows the image representing the query image for directory 23300 on the left. The image contents are small and in the top left of the image. Due to large amount of noise in the image, the York PROFI software was unable to generate image views. The image was therefore, hand edited to remove the noise as shown in the right hand image and then produced an image view.

2. In directory 23305, in marksList.txt, image R\_459013... was listed as D rather than Y (in scope) or N (out of scope). We treated this as N (out of scope).
3. In directory 23307, two of the files listed in marksList.txt were not present in the directory, one of these images 2142503... was an in scope image and the other image was out of scope. We treated these images as not present and reduced the number of images in the directory  $N$  from 57 to 55 and the number of in scope images  $n$  from 7 to 6.
4. In directory 23328, two of the files listed in marksList.txt were not present in the directory; both of these images 2166262... and 2166265 were in scope images. We treated these images as not present and reduced the number of images in the directory  $N$  from 54 to 52 and the number of in scope images  $n$  from 4 to 2.
5. In directory 23330, only one of the files listed in marksList.txt is in scope (2373748... shown in figure 14) and our view generation (blurring) was unable to generate any views from this image. The image is noisy, the contents do not cover the image, the image consists mainly of text (which is not in the scope of the PROFI project) and the lines are very jagged. We do not provide any results in the table for this query.

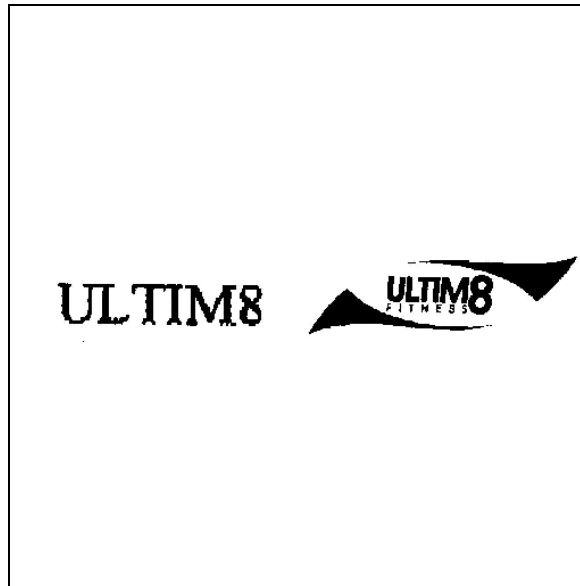


Figure 14. The PROFI software was unable to generate any image views for this image.

6. In directory 23360, one of the files listed in marksList.txt as in scope (802357... shown in figure 15) did not produce any outputs from our view generation (blurring) process. The image is noisy, the contents do not cover the image and the image consists mainly of text (which is not in the scope of the PROFI project). We do not calculate a last place accuracy figure for this query due to the missing file.



Figure 15. The original image is shown left and the greyscale PGM image created for this is shown right. The PROFI software was unable to generate any image views for this image.

7. In directory 23361, one of the files listed in marksList.txt as not in scope (707688... shown in figure 16) did not produce any outputs from our view generation (blurring) process. The image is noisy, the image did not convert well from colour to greyscale and the image consists mainly of text (which is not in the scope of the PROFI project).

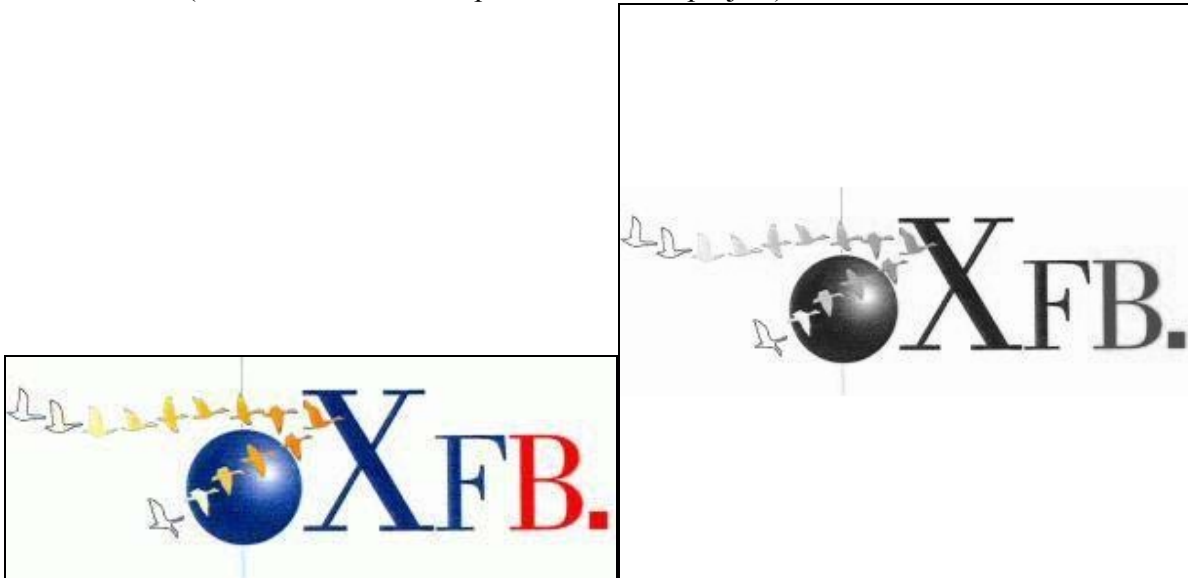


Figure 16. The original image is shown left and the greyscale PGM image created for this is shown right. The PROFI software was unable to generate any image views for this image.

8. In directory 23363, one of the files listed in marksList.txt as in scope (660627... shown in figure 17) did not produce any outputs from the shape finding process. The image is extremely noisy, and the image contains text (which is not in the scope of the PROFI project). The view generation generated views but the views contained too many shapes due to the noise in the image.

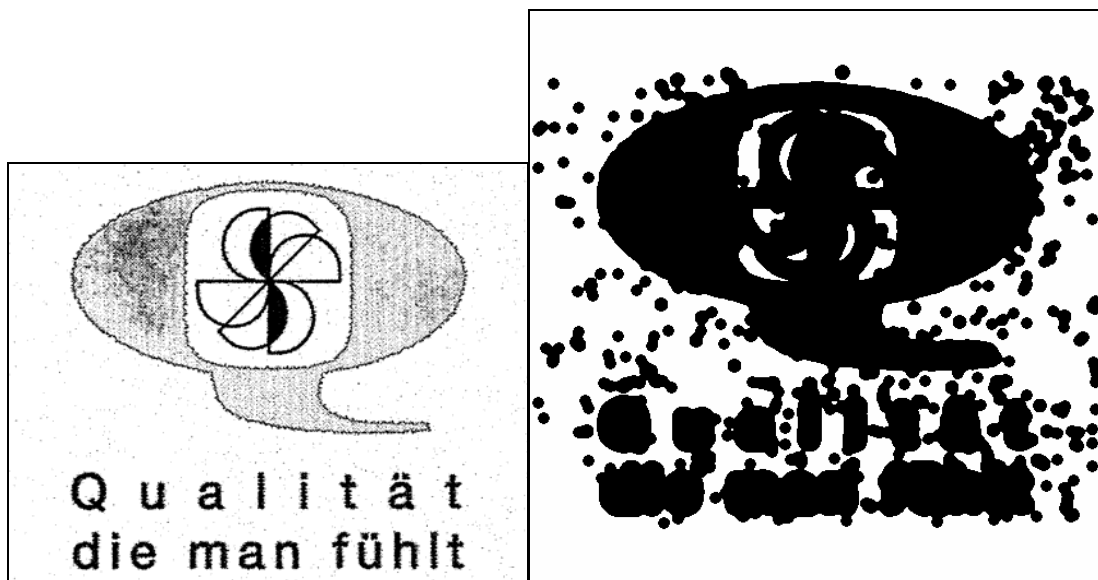


Figure 17. The original image is shown left and one of the blurred image views created for this is shown right. The PROFI software was unable to generate any shape outputs for this image as there are too many shapes to process due to the noise in the image.

9. In directory 23402, 4 images of the files listed in marksList.txt are missing. These images were all in scope. We treated the images as not present and reduced the number of images in the directory  $N$  from 176 to 172 and the number of in scope images  $n$  from 44 to 40. Two of the images did not produce any image views from the blurring process; neither of these images was in scope. One image did not produce any shapes when the shape finder processed the image views produced for that image. This image was not in scope.
10. In directory 25024, two of the files listed in marksList.txt (2028250A..., 2028250B...) were not present in the directory. Both of these images were in scope. We treated the images as not present and reduced the number of images in the directory  $N$  from 184 to 182 and the number of in scope images  $n$  from 46 to 44.
11. In directory 25212, one of the files listed in marksList.txt as in scope (776087... shown in figure 18) did not produce any outputs from our view generation (blurring) process. The image is noisy, the image did not convert from colour to greyscale well and the image consists mainly of text (which is not in the scope of the PROFi project). We do not calculate a last place accuracy figure for this query due to the missing file.



*Figure 18. The original image is shown left and the greyscale PGM image created for this is shown right. The PROFi software was unable to generate any image views for this image.*

12. In directory 25246, two of the files listed in marksList.txt (2281128..., 2281231...) were not present in the directory. Both of these images were in scope. We treated the images as not present and reduced the number of images in the directory  $N$  from 62 to 60 and the number of in scope images  $n$  from 12 to 10.
13. In directory 25266, one of the files listed in marksList.txt as in scope (MI011857... shown in figure 19) did not produce any outputs from our view generation (blurring) process. The image is noisy, the image contents do not cover the image well and the image consists mainly of text (which is not in the scope of the PROFi project). We do not calculate a last place accuracy figure for this query due to the missing file.

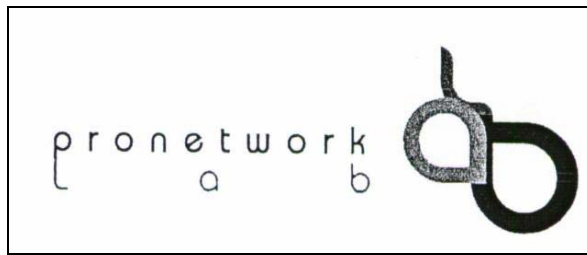


Figure 19. The PROFİ software was unable to generate any image views for this image.

14. In directory 25301, one of the files listed in marksList.txt (815650...) was not present in the directory. This image was in scope. We treated the image as not present and reduced the number of images in the directory  $N$  from 240 to 239 and the number of in scope images  $n$  from 60 to 59.
15. In directory 25306, 120 of the files listed in marksList.txt were not present in the directory, 5 of these images were in scope images and the other 115 images were not in scope images. We treated these images as not present and reduced the number of images in the directory  $N$  from 324 to 204 and the number of in scope images  $n$  from 81 to 76.
16. In directory 25336, one of the files listed in marksList.txt as in scope (2075741... shown in figure 20) and one of the files listed as not in scope (075077) did not produce any outputs from the shape finding process. The images are extremely noisy and the images contain text (which is not in the scope of the PROFİ project). The view generation generated views but the views contained too many shapes due to the noise in the image. We do not calculate a last place accuracy figure for this query due to the missing file.

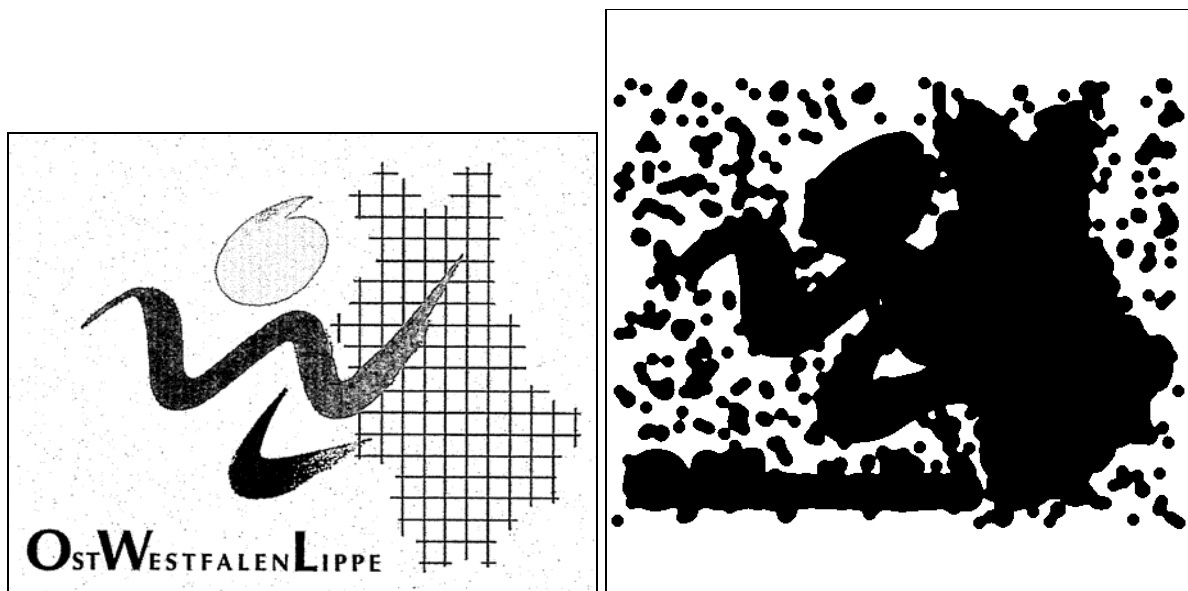


Figure 20. The original image is shown left and one of the blurred image views created for this is shown right. The PROFİ software was unable to generate any shape outputs for this image as there are too many shapes to process due to the noise in the image.

17. In directory 25339, one of the files listed in marksList.txt as not in scope (MI002379... shown in figure 21) did not produce any outputs from the shape finding process. The image is noisy, the contents do not cover the image and the image consists of text (which is not in the scope of the



PROFI project). The view generation generated views but the views contained too many shapes due to the noise in the image.

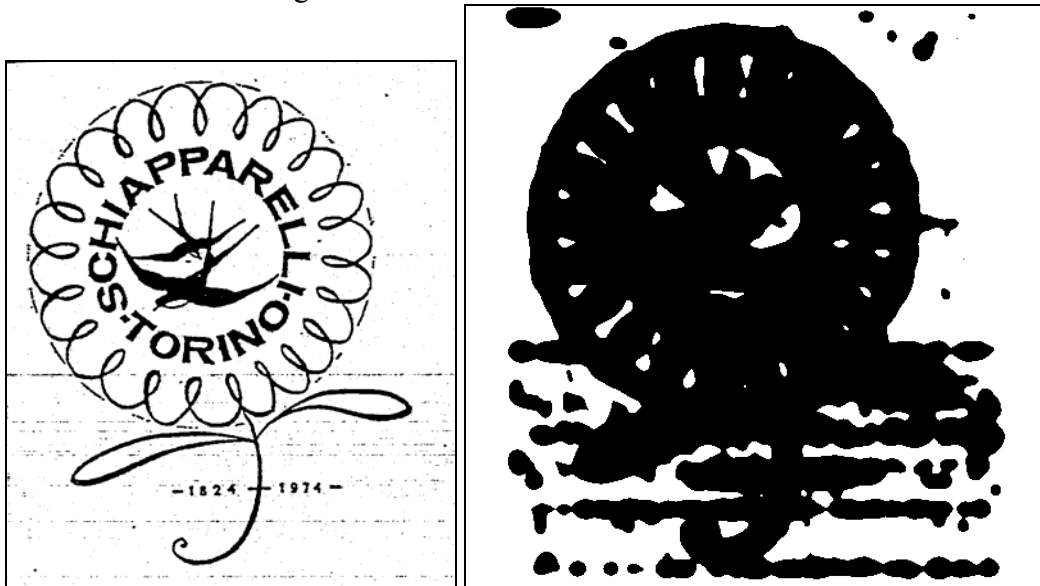


Figure 21. The original image is shown left and one of the blurred image views created for this is shown right. The PROFi software was unable to generate any shape outputs for this image as there are too many shapes to process due to the noise in the image.

18. In directory 25377, five of the files listed in marksList.txt were not present in the directory, four of these images were in scope images 2019270..., 2031345..., 2169874..., 2178269... and the other image was out of scope. We treated these images as not present and reduced the number of images in the directory  $N$  from 172 to 167 and the number of in scope images  $n$  from 43 to 39.
19. In directory 25517, two of the files listed in marksList.txt were not present in the directory; both of these images were in scope images (2378658..., 2269516...). We treated these images as not present and reduced the number of images in the directory  $N$  from 84 to 82 and the number of in scope images  $n$  from 21 to 19. One of the files listed in marksList.txt as in scope (783963... shown in figure 22) did not produce any outputs from our view generation (blurring) process. The image is noisy, the image does not convert from colour to greyscale well and the image consists mainly of text (which is not in the scope of the PROFi project). We do not calculate a last place accuracy figure for this query due to the missing file.



Figure 22. The original image is shown left and the greyscale PGM image created for this is shown right. The PROFI software was unable to generate any image views for this image.

#### 4.2.1 Score measures

We follow the evaluation procedure introduced by ARTISAN [EGB97] and calculate three measures

- Normalised Recall =  $1 - \frac{\sum_{i=1}^n R_i - \sum_{i=1}^n i}{n(N-n)}$
- Normalised Precision =  $1 - \frac{\sum_{i=1}^n (\log R_i) - \sum_{i=1}^n (\log i)}{\log\left(\frac{N!}{n!(N-n)!}\right)}$

Where there are a large number of files in a directory (i.e.,  $N > 170$ ), then  $N!$  is too large and normalised precision calculates as 1. For such query directories, we calculate the

$$\text{standard precision} = \frac{n}{R_{last}}$$

where  $R_{last}$  is the rank at which the last relevant (in scope) document *last* is actually retrieved. Where we have used this figure, we highlight the score in blue in **Table 2**

- Last Placed Ranking =  $1 - \frac{R_{last} - n}{N - n}$

Where  $R_i$  is the rank at which document *i* is actually retrieved,  $R_{last}$  is the rank at which the last relevant (in scope) document *last* is actually retrieved, *n* is the total number of relevant (in scope) documents and *N* is the size of the whole document collection.

For all measures a higher score is desirable: a score of 0 indicates failure to find the relevant images and a score of 1 indicates that the relevant documents were found in the highest ranks.

Due to the problems processing the data as noted above, we have had to accommodate the problems within the measure calculation. If files are missing from the data set then we reduce *N* and *n* accordingly. If we are unable to process an image that is in scope, either the blurring cannot produce

image views or the shape finding cannot find shapes in the views produced then we rank these images as equal last. For example, if there are 100 images in a directory and we cannot process 2 then both of these unprocessed images are ranked 99.5 where ranks range  $1...N$ .

As noted by [EGB97], these three calculated measures all estimate retrieval accuracy but each measure emphasises a different characteristic of the retrieval performance. Normalised recall describes the system performance at high ranks, normalised precision assesses performance at all ranks and the last place measure demonstrates the system's effectiveness at retrieving all images by assessing the rank of the last relevant (in scope) image.

## 5 Results

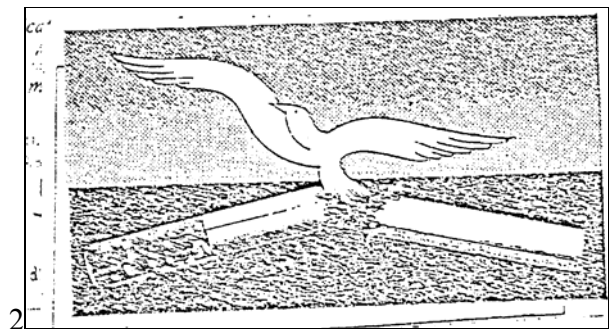
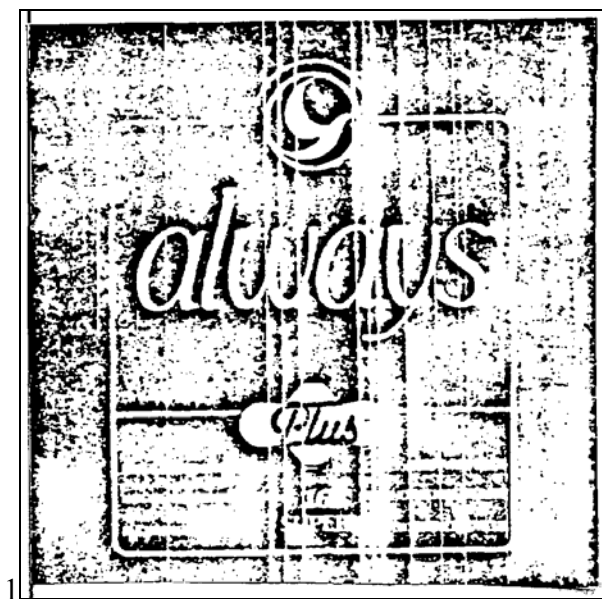
The results are shown in Table 2. The recall varies between 89% for directory 23300 and 32% for directory 23285. The images for directory 23285 are shown in appendix B and the images for directory 23300 are shown in appendix C. The recall is normalised and emphasises the accuracy of the top retrieval matches. The precision varies between 87% and 14%. (We note that we are unable to calculate the normalised precision for some queries due to the large numbers of images in the directories and use a standard precision figure for these directories instead). Normalised precision assesses the accuracy across the range of ranks. The last place varies between 0.88 and 0 and demonstrates the system’s effectiveness at retrieving all images. The unweighted average normalised recall across the 34 queries is **55%**. We feel the results are mixed.

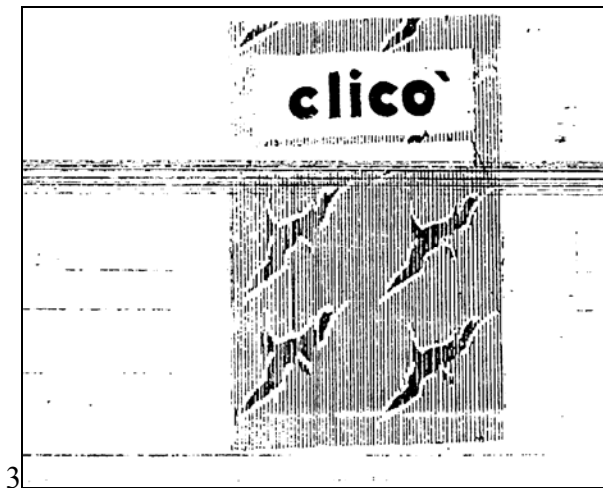
Query ID (Directory)	N	n	Recall	Precision	Last Place
23285	56	6	0.3233	0.1438	0.04
<b>23300</b>	<b>52</b>	<b>2</b>	<b>0.8900</b>	<b>0.5580</b>	<b>0.88</b>
<b>23305</b>	17	5	0.8833	0.8686	0.6667
<b>23307</b>	<b>55</b>	<b>6</b>	0.6565	0.5384	0.2245
23327	51	1	0.8000	0.3901	0.8
<b>23328</b>	<b>52</b>	<b>2</b>	0.6	0.2563	0.48
<b>23330</b>	<b>N/R</b>	<b>N/R</b>	<b>N/R</b>	<b>N/R</b>	<b>N/R</b>
23337	60	10	0.4100	0.2764	0.04
23358	76	19	0.4266	0.3228	0.1228
<b>23360</b>	<b>72</b>	<b>18</b>	0.5751	0.5263	<b>N/R</b>
<b>23361</b>	<b>52</b>	<b>2</b>	0.51	0.3371	0.1
<b>23363</b>	<b>60</b>	<b>10</b>	0.462	0.3001	<b>N/R</b>
23367	53	3	0.4000	0.1448	0.4
23379	62	12	0.4417	0.3454	0.08
<b>23402</b>	<b>172</b>	<b>40</b>	0.4763	<b>0.2367</b> <i>(n/R<sub>last</sub>)</i>	<b>N/R</b>
23415	58	8	0.5875	0.3593	0.26
<b>25024</b>	<b>182</b>	<b>44</b>	0.6653	<b>0.25</b> <i>(n/R<sub>last</sub>)</i>	0.0435
25207	53	3	0.5200	0.2657	0.08
<b>25212</b>	<b>60</b>	<b>10</b>	0.414	0.2682	<b>N/R</b>
25216	64	14	0.4686	0.3332	0.1
25244	92	23	0.6377	0.5351	0.1159
<b>25246</b>	<b>60</b>	<b>10</b>	0.48	0.3131	0.02
25260	108	27	0.5839	0.4414	0.0741
<b>25266</b>	<b>172</b>	<b>43</b>	0.5486	<b>0.25</b> <i>(n/R<sub>last</sub>)</i>	<b>N/R</b>
<b>25301</b>	<b>239</b>	<b>59</b>	0.6600	<b>0.2706</b> <i>(n/R<sub>last</sub>)</i>	0.1167
<b>25306</b>	<b>204</b>	<b>76</b>	0.5531	<b>0.3725</b> <i>(n/R<sub>last</sub>)</i>	0
25308	92	23	0.4650	0.3428	0.0435
25311	88	22	0.4731	0.3663	0

25336	208	52	0.5437	0.2506 (n/R <sub>last</sub> )	N/R
25339	176	44	0.5425	0.2558 (n/R <sub>last</sub> )	0.0303
25377	167	39	0.4944	0.3523	0.0234
25493	55	5	0.4000	0.1916	0.1
25500	63	13	0.6692	0.5037	0.22
25517	82	19	0.5873	0.4014	N/R

**Table 2.** The table lists the number of images in the database  $N$ , the number of in scope images in the database  $n$ , normalised recall, normalised precision and last place count for 34 queries each applied to its own database from the 34 databases present within the Aktor data set evaluated. N/R indicates no result. Text in red indicates where there were problems with the data set or problems processing the data set. Where the precision is listed in blue text, we have used the standard precision calculation rather than the normalised as the number of files in the directory was too large to use the normalised version.

Many of the images in the data set provided are difficult to process. Many are extremely noisy, many contain large areas of text which is not in the scope of the PROFi project, many images are coloured TIFFs and do not convert to greyscale well and some images have the image structures compressed in a small area of the image with much background. Unfortunately, this background is often noisy so up-scaling the image structure to cover the image is not possible. We have written software filters that can remove some salt and pepper noise but they need to be initiated by human inspection and inspecting 3500 images to determine where to run the filters is not feasible. Automating the filters is not feasible as many images contain texture which is adversely affected by running a filter over it and is very difficult to distinguish noise from texture automatically. From the directory 25339, the following 8 images provide a cross-section of the images within that directory to illustrate our points. It is not even clear from human inspection whether some of these images are noisy or textured. Image 2 is probably textured whereas image 5 is probably noisy. We achieved a normalised recall accuracy of 54% and a standard precision score of 26% for this query despite the image quality.

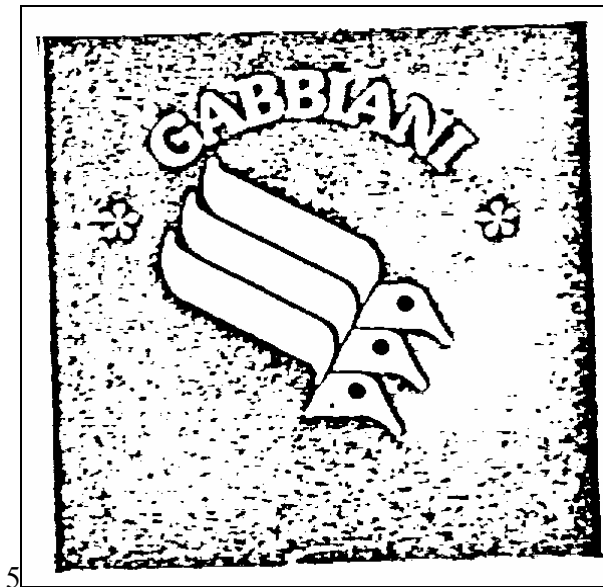




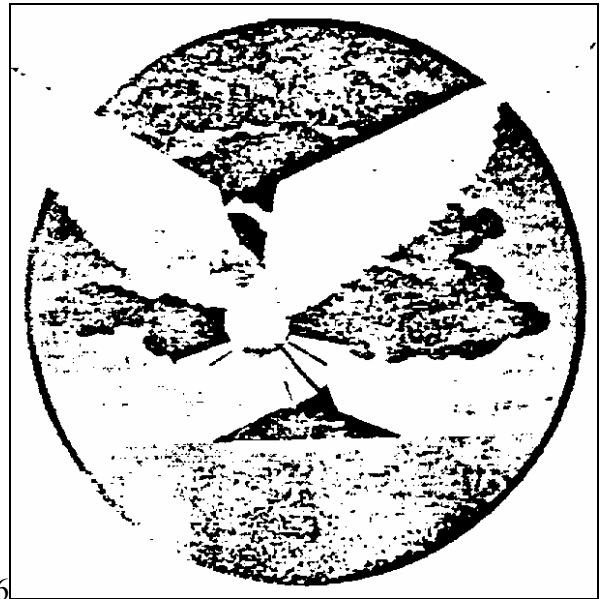
3



4



5



6

Il marchio é costituito dalla raffigurazione di uno scudo in cui predomina la figura stilizzata di un volatile ad ali aperte poggiante su un carro con ruote e con fianco a fasce inclinate; detto scudo essendo interessato inferiormente da una composizione di frutta e foglie e superiormente da un nastro ondulato ed arricciato alle estremità, comprendente la dicitura "AZIENDA AGRICOLA WALLNER" in carattere stampatello minuscolo con iniziali maiuscole, inferiormente al tutto é presente un'ulteriore dicitura "SALUS PUBLICA SUPREMA LEX" anch'essa in carattere stampatello minuscolo con iniziali maiuscole e disposta in modo arcuato.



Unfortunately, such images have an impact on the recall and precision figures as they are very difficult to process accurately. This manifests itself in the wide range of recall and precision figures in table 2.

We also feel that improvements need to be made to the graph matching framework. The original framework was developed by Alwis [A99] as a PhD thesis and was biased towards matching the images using edge segments and Alwis spent much tuning the framework to maximise recall accuracy with respect to edge segments. However, in the PROFI project, the partners have focussed on shapes more than edge segments in accordance with both the Project proposal and producing data that is useful to ALL partners. Also Alwis's original graph matching framework did not use image views to form the data. There was only a single representation of each image. The use of multiple representations introduces the problem of collating the results for each image across multiple views. We input each image view as a query and produce a set of rankings and scores for all database images against each view in turn. There are many ways to collate multiple results such as majority voting, weighted majority voting, best score etc. Hence, this change of emphasis away from edge segments to shapes and the introduction of image views is clearly impacting on the accuracy. Revising the framework fully would necessitate spending much time re-optimising the graph matching framework to rebalance the framework and rectify the bias from edge segments to shapes. We would also need to spend much time assessing the merits of the various measures for collating scores.

## 6 Conclusion

We have developed and demonstrated an image matching framework based on graph matching. The first stage is a figurative image processing pathway comprising a suite of methods to find perceptual shapes (structures) within images. Each image produces a number of views and each view produces a set of edge segments and a number of perceptual shapes. The sets of images segments and the sets of shapes found for each view may be matched using any suitable image matching framework.

In this work we perform image matching using a graph matching framework designed and implemented to identify similar images using components. Inside the graph, each graph node represents either an edge segment (image primitive) or shape/structure (perceptual object) and the graph edges represent the topological and directional relations between the node objects. The matching framework uses the Binary version of the continuous Bayesian Relaxation by Elimination framework ([TA98]). This framework comprises two main stages: making initial hypotheses about possible matches; and, improving initial hypotheses using contextual information.

The graph-matching framework has been used in conjunction with the outputs from the processing pipeline to produce an image matching system. The images matching system has been used to match query images against stored database images using data supplied to the project by Aktor.

[ESV07] stated that a trademark retrieval system should satisfy the following criteria:

- *“One should take into account every possible interpretation of a trademark image.”*
- *“It should be possible to search in big sets of images with an acceptable speed (relatively short delivery times).”*
- *“Very similar (to the query image) images in the database can not be missed (zero tolerance).”*
- *“Trademark images should be compared in great detail (such as shape, contour, and structure) taking into account all sorts of transformations (such as rotation, scaling, inversion, and blurring).”*

[ESV07] also stated that the system should produce “image interpretations”. They note that a trademark retrieval system should be able to identify image components such as shapes, regions, texture, colour and text components and should couple this with human-perception-based segmentation to find more implicit shapes and structures in the image. Both [E00] and [BSBKF99] posit that human image matching operates on multiple levels. We have replicated these suggestions in our system, both in terms of the image representation where we use blurring and categorisation to produce multiple views of images to allow perceptual shapes and structures to be identified at different levels, and in terms of image matching, where the images are matched using both primitives (CCSs) and higher level perceptual shapes/structures. The evidence from both levels (CCSs + shapes/structures) is combined during matching to produce the overall match score between two images.

The retrieval results have been mixed. This may be explained in part by problems with the data with many noisy images, images that did not convert from colour to greyscale well and many images that contained large areas of text. However, we feel there are still many improvements that could be made to the graph matching framework. The original framework was geared towards matching the edge segments and much time has been spent tuning the framework to maximise recall accuracy. However, in PROFi we have focussed on shapes more than edge segments in accordance with the Project proposal. Hence, this change of emphasis is impacting on the accuracy and would necessitate spending much time re-optimising the graph matching framework to rebalance the framework and rectify the bias



from edge segments to shapes. We have also introduced multiple representations for each image which have to be accommodated into the matching process and would benefit from further evaluation.

## References

- [A95] J. Ashley, et al, Automatic and Semiautomatic Methods for Image Annotation and Retrieval in QBIC, Proc Storage and Retrieval for Image and Video Databases Conf, 1995.
- [A99] S. Alwis. Content-Based Retrieval of Trademark Images, PhD Thesis, Dept. of Computer Science, University of York, UK, 1999
- [AA99] S. Alwis, and J. Austin. Trademark image retrieval using multiple features. Presented at CIR-99: The Challenge of Image Retrieval, Newcastle-upon-Tyne, U.K., Feb. 1999.
- [BA83] P.J. Burt & E.H. Adelson. The Laplacian Pyramid as a compact image code, IEEE Trans on Communications, 31(4):532-540, 1983.
- [BSBFK99] I Biederman, et al. Subordinate-Level Object Classification Re-examined. Psychological Research, 62:131-153, 1999.
- [CS99] G. Ciocca and R. Schettini. Similarity Retrieval of Trademark Images. In Proc. of the Intl. Conf. on Image Analysis and Processing, pp. 915-920, 1999.
- [DP97] A. Del Bimbo and P. Pala. Visual Image Retrieval by Elastic Matching of User Sketches. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI), 19(2):121-132, 1997.
- [E00] J.P. Eakins, Trademark image retrieval - a survey, Multimedia Storage and Retrieval Techniques - State of the Art (Berlin: Springer-Verlag, 2000).
- [EGB97] J.P. Eakins, M.E. Graham and J.M. Boardman. Evaluation of a trade-mark retrieval system. in *19th BCS IRSG Research Colloquium on Information Retrieval*, Robert Gordon University, Aberdeen, 1997.
- [ERE03] J. P. Eakins, K. J. Riley, J. D. Edwards, Shape Feature Matching for Trademark Image Retrieval. In: E.M. Bakker et al. (Eds.): CIVR 2003, LNCS 2728, pp. 28-38, Springer-Verlag, 2003.
- [ESB96] J. P. Eakins, K. Shields and J. Boardman. ARTISAN – a shape retrieval system based on boundary family indexing. In Proc. SPIE: Storage and Retrieval for Still Image and Video Databases, Vol. 2670, pp. 17-28, 1996.
- [ESV07] J.P. Eakins, J. Schietse, R.C. Veltkamp. Practice and Challenges in Trademark Image Retrieval. Procs 6<sup>th</sup> ACM International Conference on Image and Video Retrieval, CIVR'07, 9-11 July 2007.
- [F03] J. French, et al, An Exogenous Approach for Adding Multiple Image Representations to Content-Based Image Retrieval Systems, Proc 7th Int'l Symposium on Signal Processing and its Applications, Paris, 2003.
- [G72] E. Goldmeier. Similarity in Visually Perceived Forms, Psychological Issues, 8(1), 1972.
- [HEA06] V.J. Hodge, J. Eakins & J. Austin. Eliciting Perceptual Ground Truth for Image Segmentation. Technical Report YCS 401(2006), Department of Computer Science, University of York.
- [HEA07] V.J. Hodge, J. Eakins & J. Austin, Inducing a Perceptual Relevance Shape Classifier, Proc 6<sup>th</sup> ACM Int'l Conf. on Image and Video Retrieval, (CIVR07), Amsterdam, 2007.
- [HHEA06] V.J. Hodge, G. Hollier, J. Eakins & J. Austin, Eliciting Perceptual Ground Truth for Image Segmentation, Proc Int'l Conf on Image and Video Retrieval (CIVR06), Tempe, AZ, 2006.
- [JV98] A. K. Jain and A. Vailaya. Shape-Based Retrieval: A Case Study with Trademark Image Databases. Pattern Recognition, 31(9):1369-1390, 1998.
- [KK98] Y.S. Kim & W.Y. Kim. Content-based trademark retrieval system using a visually salient feature. Image and Vision Computing, 16:931-939, 1998.
- [KS91] P.J. Kellman and T.F. Shipley, A theory of visual interpolation in object perception, *Cognitive Psychology* **23** (1991), pp. 141–221.

- [L98] T. Lindeberg, Feature Detection with Automatic Scale Selection, *Int'l Journal of Computer Vision*, 30(2), 1998.
- [LC98] C-S. Lu & P-C. Chung, Wold Features for Unsupervised Texture Segmentation, *Proc 14th Int'l Conf on Pattern Recognition (ICPR'98)*, 1998.
- [LDH07] R. van Leuken, F. Demirci, V.J. Hodge et al, Layout Indexing of Trademark Images, *Proc ACM Int'l Conf. on Image and Video Retrieval (CIVR07)*, Amsterdam, 2007.
- [MJ92] J. Mao & A.K. Jain, Texture classification and segmentation using multiresolution simultaneous autoregressive models, *Pattern Recognition*, 25:173-188, 1992.
- [MRC98] R. Manmatha, S. Ravela, and Y. Chitti. On computing local and global similarity in images. In *Proc. SPIE Conf. on Human and Electronic Imaging III*, 1998
- [REB00] M. Ren, J.P. Eakins & P. Briggs, Human perception of trademark images: implications for retrieval system design, *Journal of Electronic Imaging*, 9(4):564-575, 2000.
- [S03] E. Saund, Finding Perceptually Closed Paths in Sketches and Drawings, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(4):475-491, 2003.
- [TA98] M Turner, J Austin. Graph matching by neural relaxation. *Neural Computing & Applications* 7(3): 238-248, Springer, 1998.
- [U93] M. Unser, A. Aldroubi & M. Eden, B-Spline Signal Processing, *IEEE Trans on Signal Processing*, 41(2) 1993, 821-833 (part I) & 834-848 (part II).
- [WB91] D.M. Wuescher & K.L. Boyer, Robust contour decomposition using a constant curvature criterion, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(1):41-51, 1991.

# Appendix A

## 7 File Formats in use by the Trademark Matcher

### 7.1 Database Files

The York PROFI software outputs the blurred image views in PGM format. These are converted to BMP format using the ImageMagick convert utility. The York processing pipeline produces two files per image view for the graph matcher: a local file which contains details of the CCSs and their relationships and a global file which contains details of the shapes, their relationships and the relationships between eh shapes and the CCSs (i.e., which CCSs are used to construct each shape).

- \* imageX\_local\_raw – one file per image view in the database
- \* image\_global\_raw – one file per image view in the database

The query application also creates local and global raw feature files for the query image views.

The database of image view files (BMP files) are indexed within the graph matcher software and each is given an image ID.

#### 7.1.1 imageX\_local\_raw

This file contains details of the CCSs for an image view. One file is created for each image view in the database. One file is also constructed for each of the query image views. The first line is the image ID.

```
Image_ID                                     <=ID
type angle ratio seg_id seg_id              <=corners
9999 9999 9999 9999 9999
seg_id type length orientation curvature perimeter <=segments
9999 c 9999 9999 9999 9999
seg_id [neighbour_N]+ #                     <=proximity
$$$$
seg_id [neighbourN]+ #                       <=collinearity
$$$$
```

An example being ...

```
182                                         <= ID
1 2.132353 1.374878 1 14                   <= corners
1 3.866667 1.236727 3 4
2 156.388657 1.585456 5 4
1 3.866667 1.236727 3 4
2 103.555023 1.199970 6 5
2 156.388657 1.585456 5 4
2 45.153702 3.363724 6 7
2 109.807549 1.199970 6 5
2 45.153702 3.363724 6 7
2 67.821167 1.095631 10 9
2 72.515587 1.095631 10 9
1 3.761905 1.295176 12 13
1 1.079365 1.562608 14 13
1 3.761905 1.295176 12 13
1 1.079365 1.562608 14 13
0 9.819305 1.042198 15 19
9999 9999 9999 9999 9999
1 c 53.823788 0.000000 145 55.426418      <= segments
2 l 5.656854 45.000011 999999 0.000000
3 c 62.177166 0.000000 116 65.598007
4 c 45.276924 0.000000 30 53.041634
5 l 84.095184 154.653824 999999 0.000000
```

```

6 c 91.235954 0.000000 373 100.911720
7 l 30.000000 89.999977 999999 0.000000
8 l 11.401754 127.874977 999999 0.000000
9 l 61.098282 21.104841 999999 0.000000
10 c 62.241467 0.000000 326 66.941147
11 l 10.630146 48.814083 999999 0.000000
12 c 34.014702 0.000000 237 33.414215
13 c 25.806976 0.000000 63 25.798986
14 c 38.052597 0.000000 68 40.313713
15 l 24.596748 116.565041 999999 0.000000
16 l 24.596748 116.565041 999999 0.000000
17 c 24.207438 0.000000 161 23.798986
18 c 5.000000 0.000000 39 3.828427
19 l 23.600847 126.384346 999999 0.000000
20 l 8.000000 -0.000000 4 0.000000
21 l 5.385165 158.198578 2 0.000000
22 l 9.433981 147.994614 999999 0.000000
23 l 4.123106 165.963760 2 0.000000
24 l 6.403124 38.659817 2 0.000000
9999 c 9999 9999 9999 9999

```

*<= proximity*

```

1 2 14 #
2 3 20 1 #
3 4 2 20 #
4 5 3 #
5 6 4 #
6 7 5 #
7 8 6 #
8 9 7 #
9 10 8 #
10 11 9 #
11 12 10 #
12 13 11 #
13 14 12 #
14 1 13 #
15 18 19 #
17 18 #
18 15 19 17 #
19 20 15 18 #
20 19 2 3 #
21 22 #
22 21 #
$$$$

```

*<= collinearity*

```

1 17 #
15 19 #
17 1 #
19 15 #
21 22 #
22 21 #
$$$$

```

## 7.1.2 imageX\_global\_raw

One file is built for each image view in the database and contains the details of the shapes, their relationships and the relationships between the CCS and the shapes (which CCSs are used to construct each shape). One file is also constructed for each query image view.

```

Image_ID
hull_circumference (unused)
Figure_ID circumference hull_circumference directionality [seg_id]+ # <=Shape feature vector
$$$$
Figure [rel_angle]+ # <=Shape relations

```

an example being . .

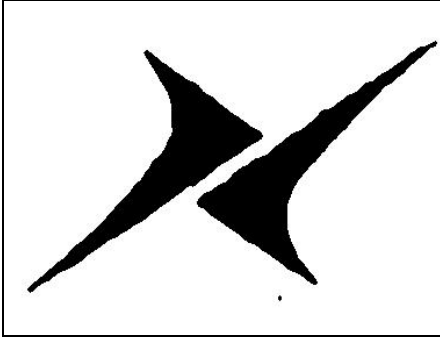
```
172
0.000000
1 0.558252 1288084.125000 0.248115 1 7 6 5 4 3 2 # <=Shape feature vector
2 0.268201 918413.000000 0.254552 8 23 22 21 20 19 18 17 14 13 12 11 10 9 #
3 0.236382 897464.875000 0.041051 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 #
4 0.416408 399667.062500 0.085376 24 33 32 31 30 29 28 27 26 25 #
$$$$
1 0.000000 8.688454 8.545727 17.926044 # <=Shape relations
2 8.688454 0.000000 0.142727 26.614498 #
3 8.545727 0.142727 0.000000 26.471771 #
4 17.926044 26.614498 26.471771 0.000000 #
```

## Appendix B

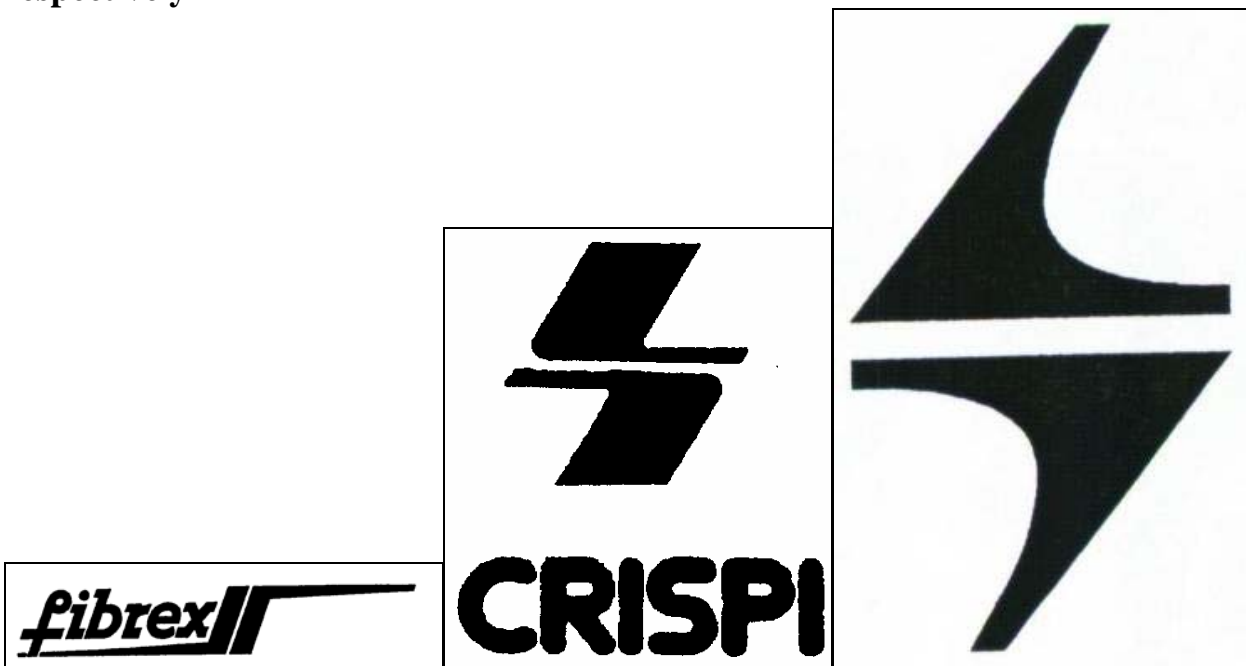
### 8 The unprocessed images for Query 23285. (Query with lowest normalised recall)

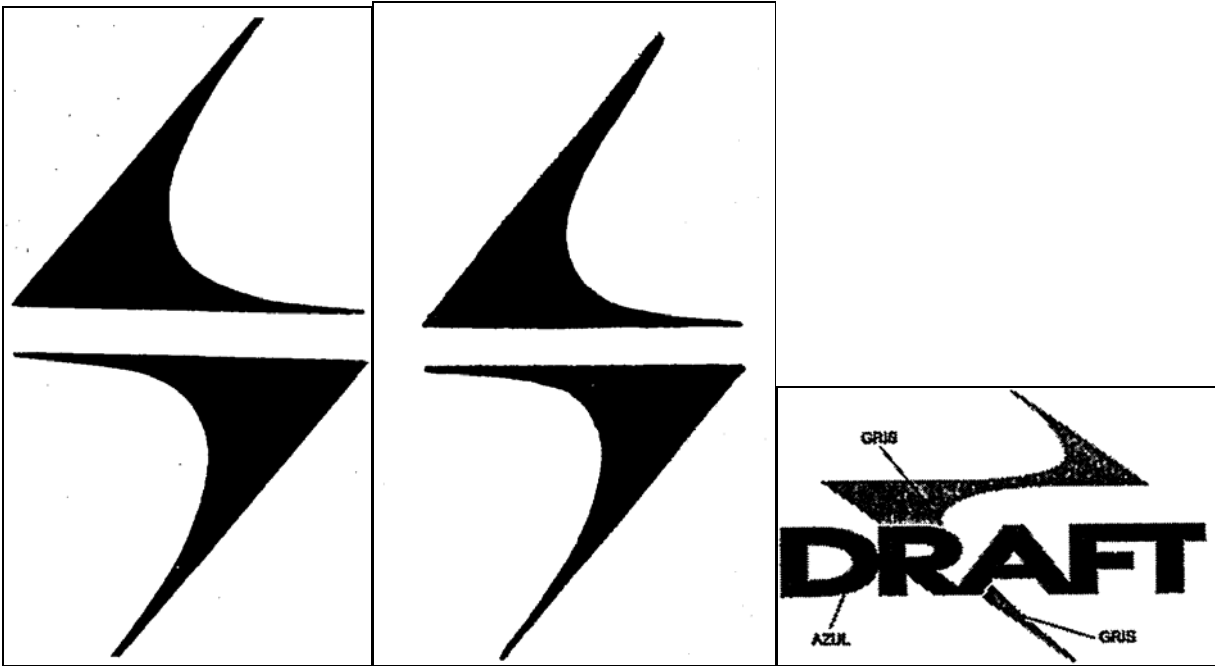
Prior to view generation, each image is padded with white space to make it square and resized to 512x512 pixels.

#### Query Image

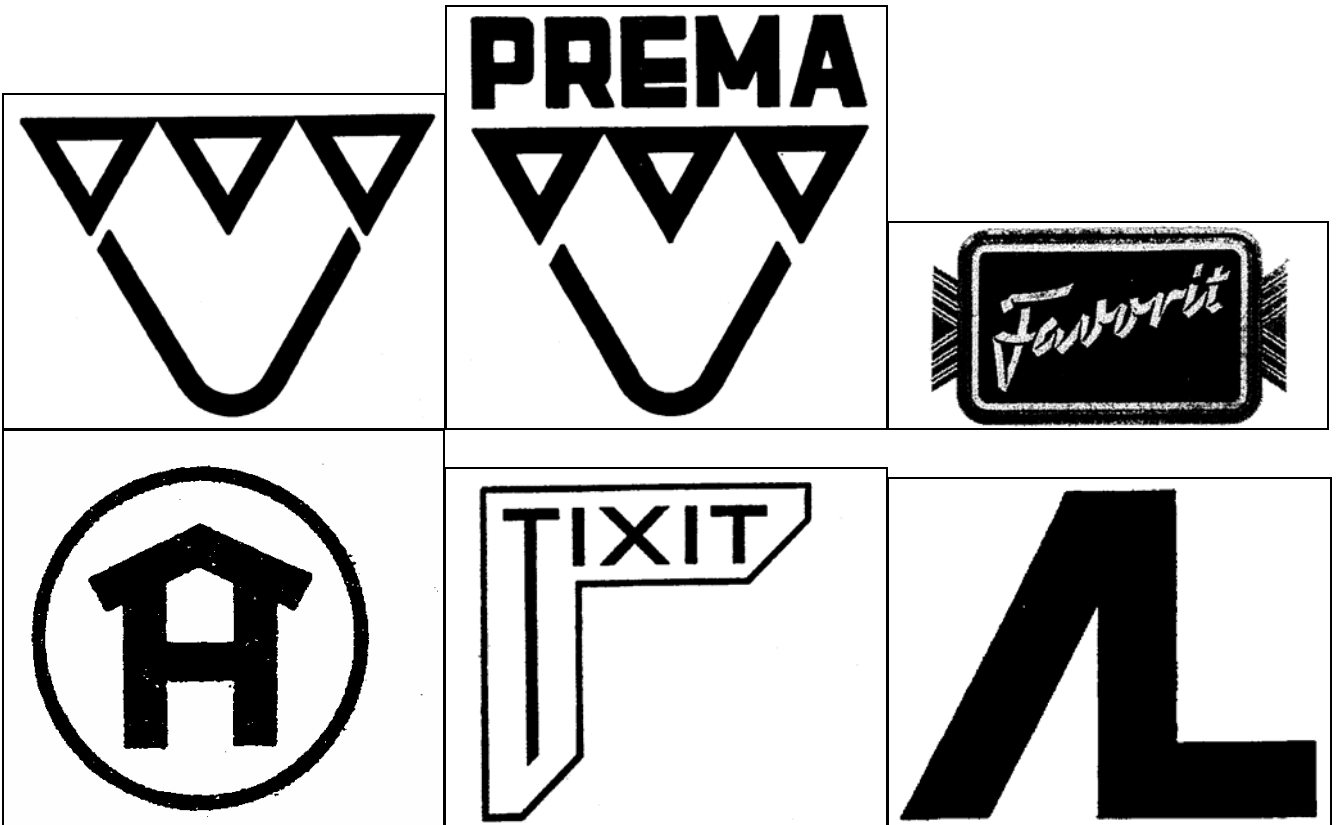


In scope Database Images (6) – ordered by rank and ranked 26, 32, 35, 43, 44, 54 respectively

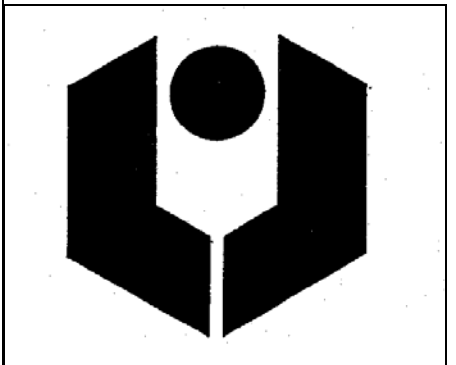
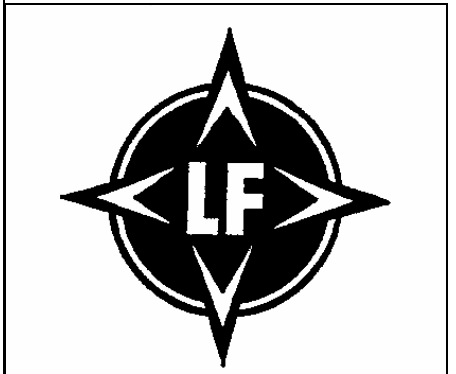
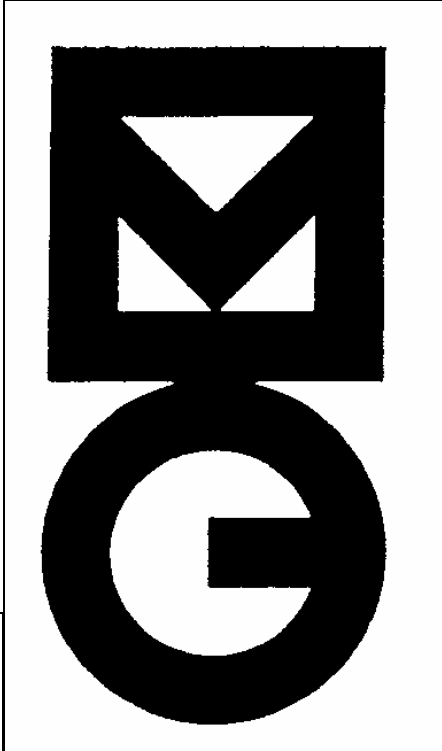
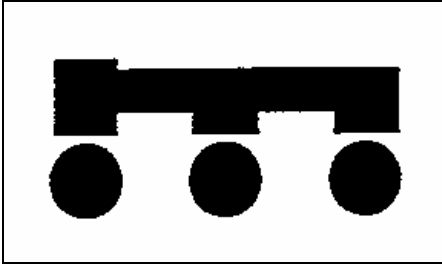


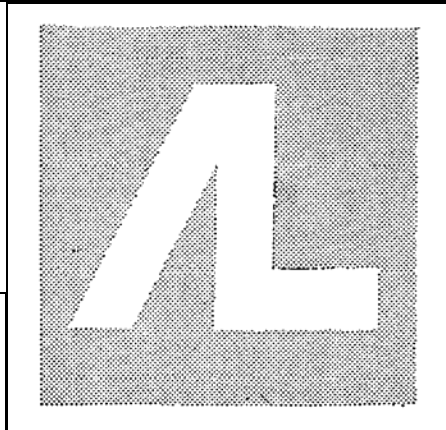
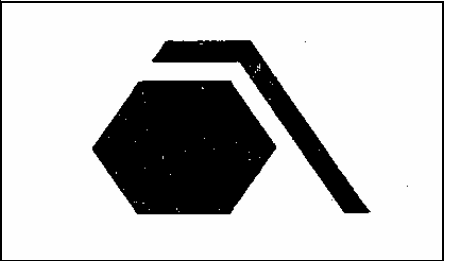
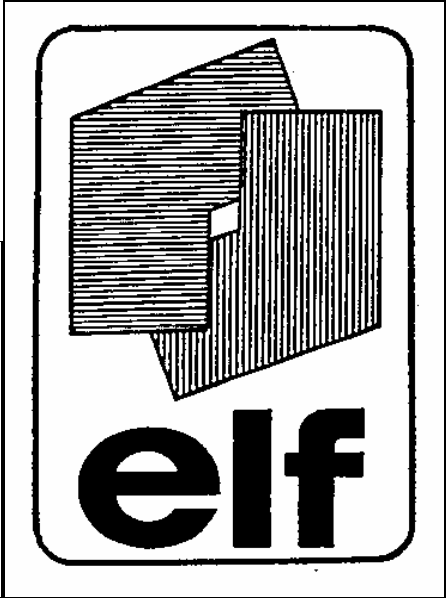
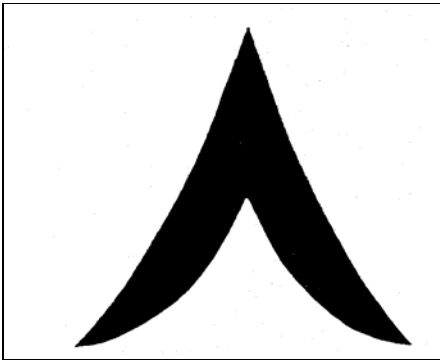
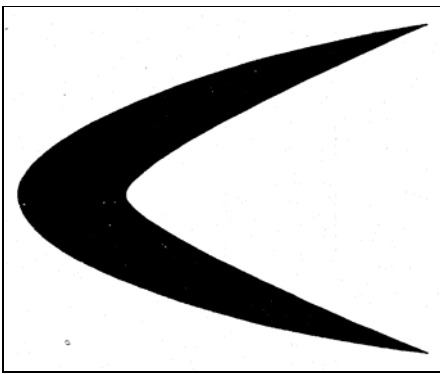


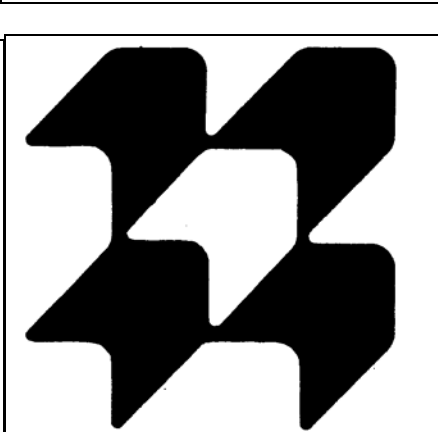
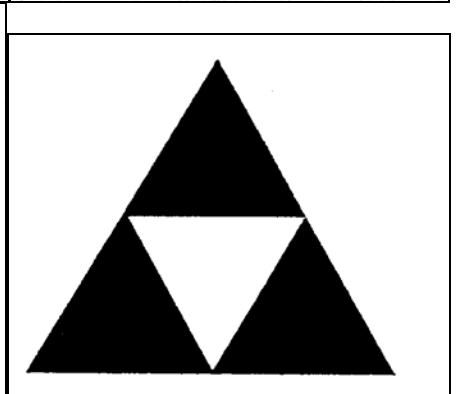
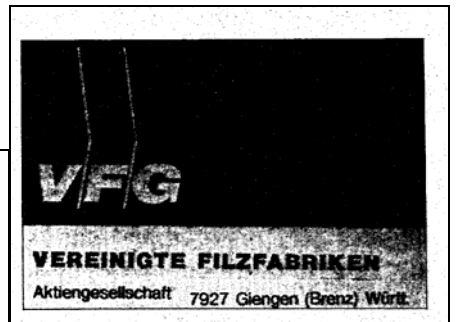
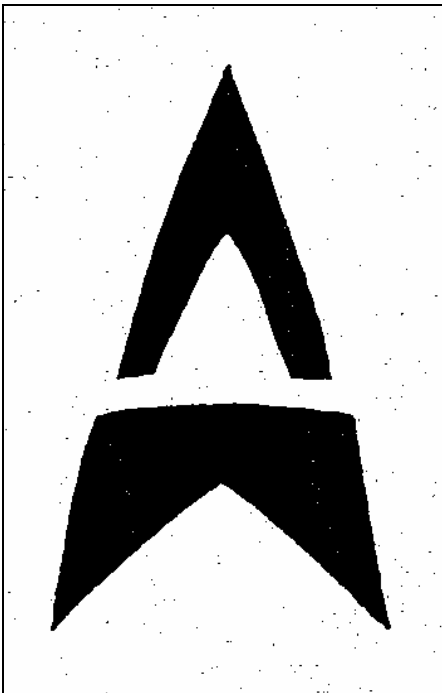
Out of scope Database Images (50) (not ordered)

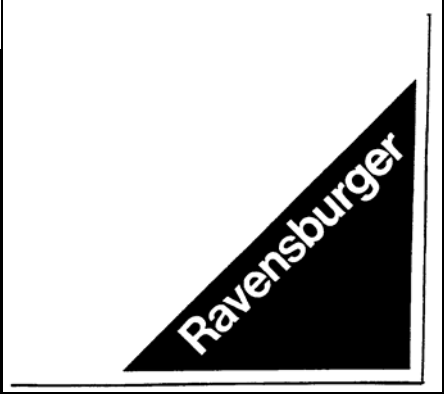
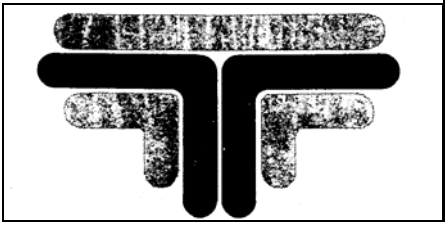
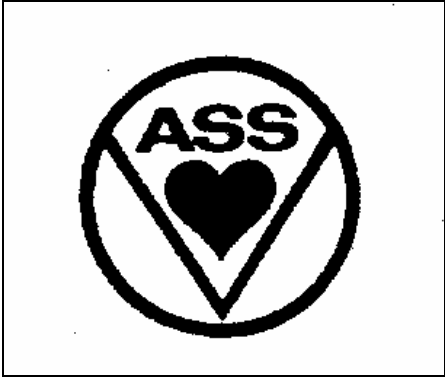


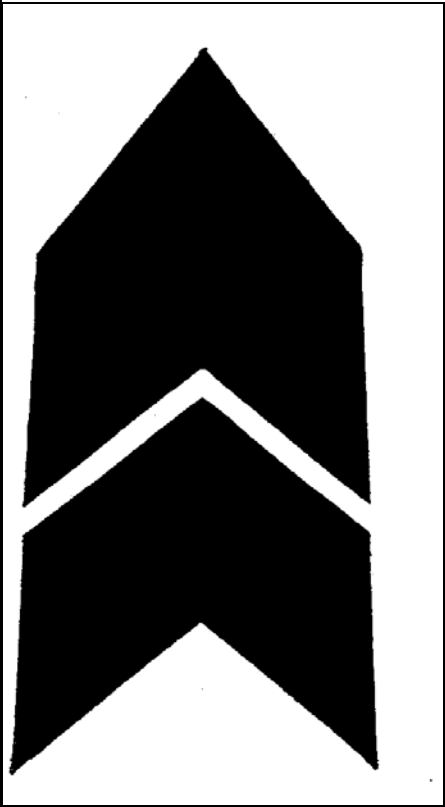
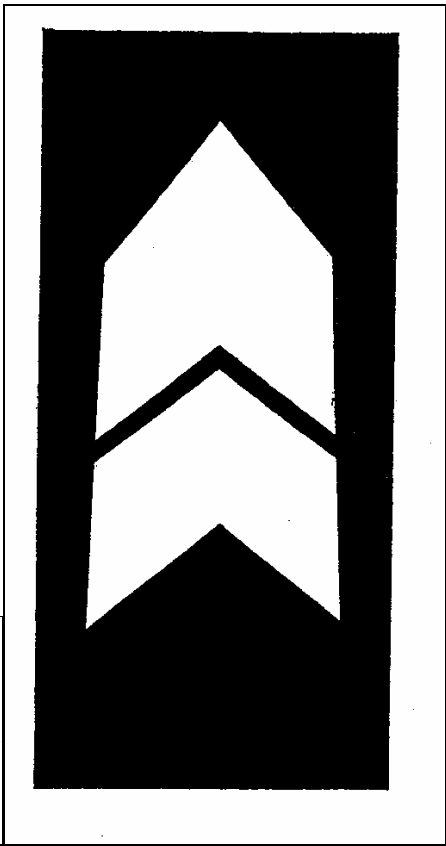
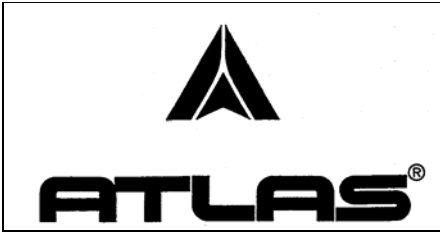


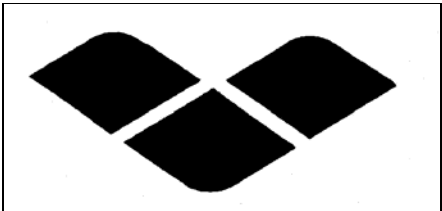
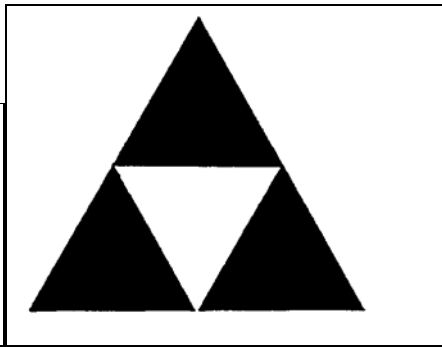
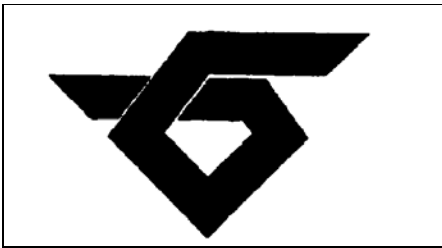










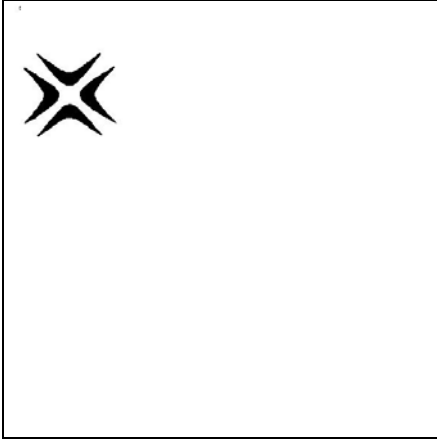


## Appendix C

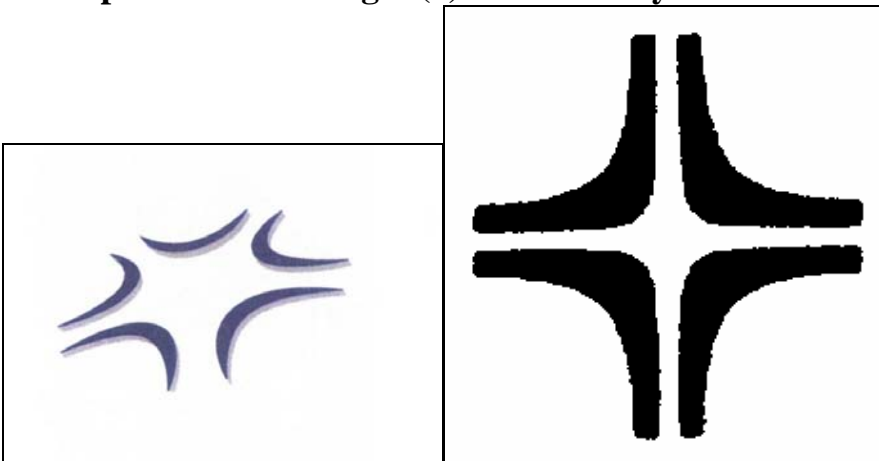
### 9 The unprocessed images for Query 23300. (Query with highest normalised recall)

Prior to view generation, each image is padded with white space to make it square and resized to 512x512 pixels.

#### Query Image



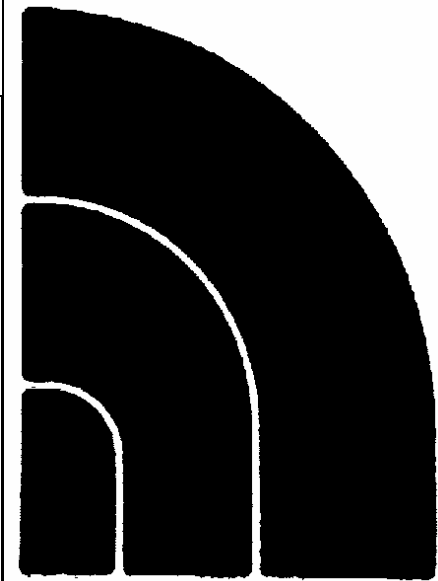
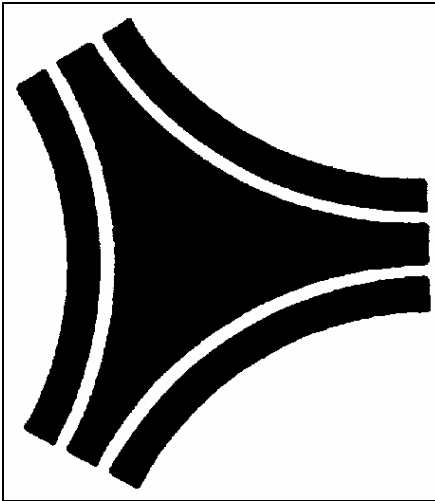
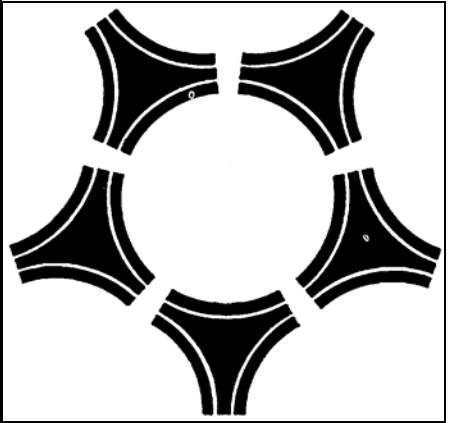
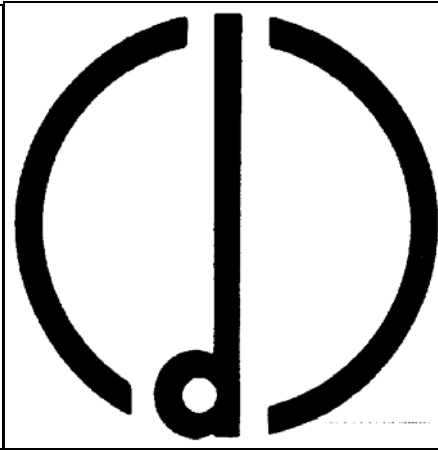
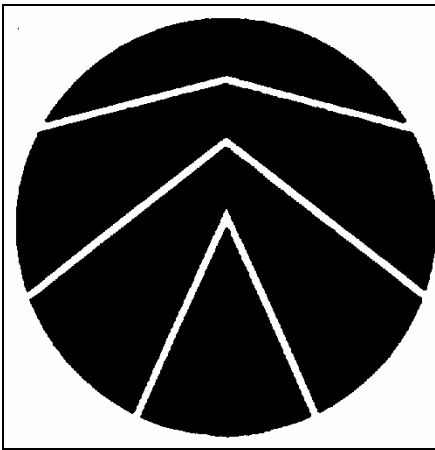
#### In Scope Database Images (2) - ordered by rank and ranked 6 and 8 respectively

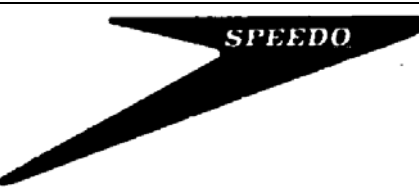
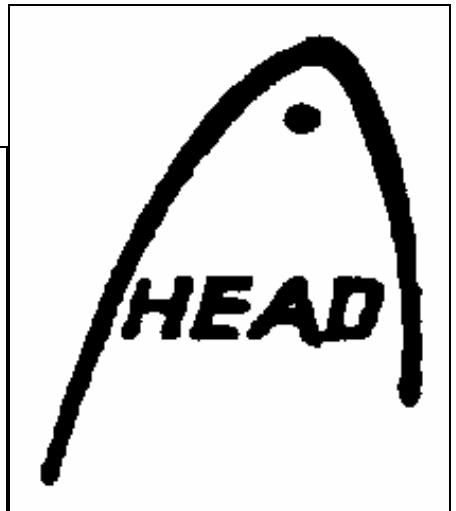
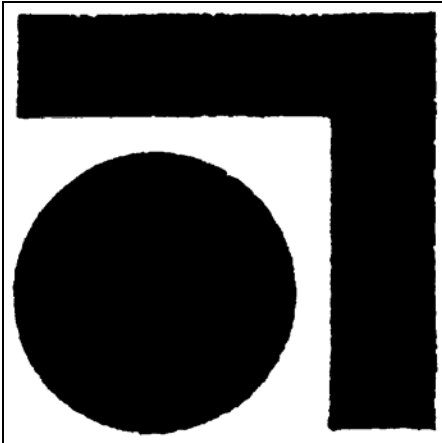


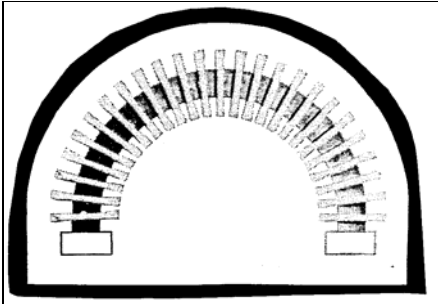
Out of Scope Database Images (50) (not ordered)











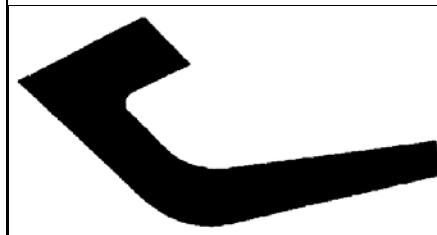
**sponeta**



**Dagrofa**



**Hoki**

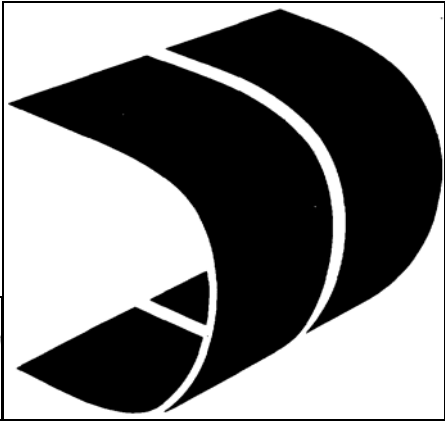


**BRIDGESTONE**



**robbe**

**SuperCar**  
BIL & FRITID



**FUKUDA**

**Danfill**

**k**  
**kawasaki**

