The Trace transform and its applications

A Kadyrov and M Petrou [*]
School of Electronic Engineering,
Information Technology and Mathematics,
University of Surrey,
Guildford, GU2 7XH, UK

Abstract

The Trace transform proposed, a generalisation of the Radon transform, consists of tracing an image with straight lines along which certain functionals of the image function are calculated. Different functionals that can be used may be invariant to different transformations of the image. The paper presents the properties the functionals must have in order to be useful in three different applications of the method: construction of invariant features to rotation, translation and scaling of the image, construction of sensitive features to the parameters of rotation, translation and scaling of the image, and construction of features that may correlate well with a certain phenomenon we wish to monitor.

Key words: Radon transform, Trace transform, Invariant features, Image database search, Change detection.

1 Introduction

It has been known for some time that a 2D function can be fully reconstructed from the knowledge of its integrals along straight lines defined in its domain. This is the well known Radon transform [7] which has found significant application recently in computerised tomography [15]. A derivative of the Radon transform is the Hough transform which is a variation appropriate for sparse images like edge maps [6]. The Trace transform we are proposing here is only similar to the Radon transform in the sense that it also calculates functionals of the image function along lines criss-crossing its domain. The Radon transform calculates a specific functional, namely the integral of the function. The Radon transform, therefore, is a special case of the Trace transform. With the Trace transform, the image is transformed into another "image" which is a 2D function depending on parameters (ϕ, p) that characterise each line. Figure 1 defines these parameters. Parameter t is defined along the

[*]Corresponding author

line with its origin at the foot of the normal. As an example, figure 2 shows an image and its Trace transform [1], which is a function of (ϕ, p). We may further compute a functional of each column of the Trace transform, (ie along the p direction) to yield a string of numbers that is a function of ϕ. Finally, we may compute a third functional from this string of numbers, to yield a single number that, with the appropriate choice of the functionals we use, may have some properties we desire.

In this paper we shall discuss ways by which each of these functionals may be chosen, so that the calculated number has one of the following properties:

- is invariant to rotation, translation and scaling;
- is sensitive to rotation translation and scaling so that these transformation parameters between two images can be recovered;
- correlates well with some property we wish to identify in a sequence of images.

The above three classes of features may find direct applications to three major areas of research:

- The search of an inventory type database where the objects depicted may differ by rotation translation and scaling from the query object that has to be identified.
- The identification of transformation parameters between an inspected image and a reference one, so that the two images may be brought into registration for the purpose of direct comparison for fault or fraud detection.
- The identification of trends in a series of images, using features that correlate well with such trends, for the purpose of change detection, site monitoring and surveillance.

We shall present some example applications of the Trace transform to all these three areas.

As a tool to invariant feature construction, the Trace transform is related to Integral Geometry [10, 8] and to that class of approaches to Algebraic invariance, which integrate over the group of transformations of an image. However, it is more general than either of the above, as it does not

[1] In this example the functional computed along each tracing line was the median of the function weighted by the modulus of its first derivative. Let us denote by t_i the n sampling points along a tracing line at which the first derivative $(f'(t))$ of the image function $(f(t))$ is not 0. We rank these points in order of increasing image function $f(t)$: $f(t_1) \leq f(t_2) \leq \ldots \leq f(t_n)$. We choose t_p so that $\sum_{i=1}^{p} |f'(t_i)| = \sum_{i=p+1}^{n} |f'(t_i)|$. If such an index exists, the median of the function weighted by the modulus of its first derivative is $\frac{f(t_p)+f(t_{p+1})}{2}$. If no such index exists, we choose the largest index p which makes $\sum_{i=1}^{p-1} |f'(t_i)| < \sum_{i=p}^{n} |f'(t_i)|$. Then the median of the function weighted by the modulus of its first derivative is $f(t_p)$.

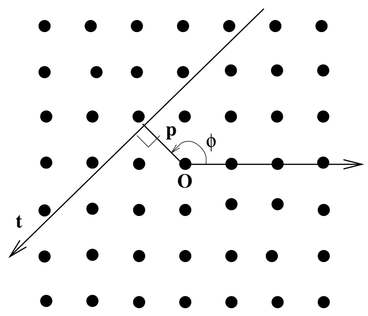


Figure 1: Definition of the parameters of an image tracing line

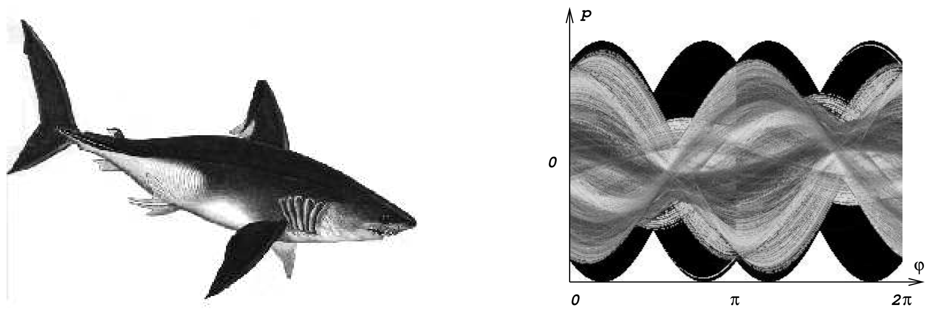


Figure 2: An image and its Trace transform

just integrate; it performs instead the calculation of any functional over the group of transformations of the image. The closest work in this area is that by Schael and Burkhardt [12, 13] who use local windows to construct features invariant to rotation and translation: Inside each local window they compute a monomial function of the grey values of the pixels, ie a function that is the product of grey values at specific positions inside the window, raised at some combination of exponents. Then they rotate the image inside the window, re-sample and recalculate the same function. This is repeated for many rotation angles. At the end they average over all rotations used. This is equivalent to integrating over all possible rotations. The same is repeated over many such local windows in the image and all results are again averaged over all window translations. This is equivalent to integrating over all possible translations. The number that results is a feature that characterises the image in a rotation and translation invariant way. The work we present here is much more general, as we do not simply integrate over rotations and translations, but we use a variety of functionals instead. The work presented by Schael and Burkhardt has the advantage over our approach that it uses local windows. It therefore allows the recognition of parts of objects. Our global approach works only for full objects. However, our global transform offers a framework for a large number of possibilities, since it may not only be used for invariant feature construction, but for other applications as well.

Another advantage of our framework, is that it offers the possibility of constructing thousands of features. Indeed, in order to be able to recognise objects, apart from the issue of invariant object descriptors, one has also the problem of the large number of features necessary. The more classes one has to discriminate, the more features may be necessary. Therefore, a mechanism is needed, capable of generating a large number of invariant features, which do not necessarily have physical or geometric meaning. The theory we shall present in the next section is general and has exactly this property. It can be used for grey images subject to linear distortions, and the framework actually can also be generalised to higher complexity deformations, although the solution to the problem is not straightforward.

In section 2 we shall present the basic tool we are proposing, namely the *Trace transform* and the way we construct from it *triple features*. In section 3 we shall discuss the way we can construct invariant features. In section 4 we shall discuss the issue of recovering the transformation parameters, and in section 5 we shall discuss the issue of solving a completely different problem, namely that of change detection. We shall present the computational complexity of the approach in section 6 and our conclusions in section 7.

2 The theory of triple feature extraction

Let us start by assuming that an object is subject to linear distortions: rotation, scaling and translation. This is equivalent to saying that the image remains the same but it is viewed from a linearly distorted coordinate system. We can understand this statement as follows: Let us call the original coordinate system of the image C_1 and the distorted C_2. C_2 is obtained from C_1 by rotation by angle $-\theta$, scaling of the axes by parameter ν and by translation with vector $(-s_0 \cos\psi_0, -s_0 \sin\psi_0)$. Now let us suppose that we have a 2D image F which is viewed from C_1 as $F_1(x, y)$ and from C_2 as $F_2(\tilde{x}, \tilde{y})$. $F_2(\tilde{x}, \tilde{y})$ can be considered as an image constructed from $F_1(x, y)$ by rotation by θ, scaling by ν^{-1} and shifting by $(s_0 \cos\psi_0, s_0 \sin\psi_0)$.

It can be shown that in the new coordinate system straight lines will still appear as straight lines. That is, linear transformations preserve lines, ie an image is gliding along lines when it is linearly transformed. If a line in C_2 is parameterised by (ϕ, p, t), in the old coordinate system, C_1, its parameters are:

$$\phi_{old} \;=\; \phi - \theta \tag{1}$$

$$p_{old} = \nu \left[p - s_0 \cos(\psi_0 - \phi)\right] \tag{2}$$

$$t_{old} = \nu \left[t - s_0 \sin(\psi_0 - \phi)\right] \tag{3}$$

Consider scanning an image with lines in all directions. Denote by Λ the set of all these lines. The Trace transform is a function g defined on Λ with the help of T which is some functional of the image function when it is considered as a function of variable t. We call T the *trace functional.* If $L(C_1; \phi, p, t)$ is a line in coordinate system C_1, then:

$$g(F; C_1; \phi, p) = T(F(C_1; \phi, p, t)) \tag{4}$$

where $F(C_1; \phi, p, t)$ means the values of the image function along the chosen line. Taking this functional we eliminate variable t. The result is a two dimensional function of the variables ϕ and p and can be interpreted as another image defined on Λ.

We define a *triple feature* with the help of two more functionals designated by letters P and Φ and called *diametrical* and *circus functionals* respectively. P is a functional of the Trace transform function when it is considered as a function of the length of the normal to the line only. Φ is a functional operating on the orientation variable, after the previous two operations have been performed. Thus, the triple feature Π is defined as:

$$\Pi(F, C_1) = \Phi(P(T(F(C_1; \phi, p, t)))) \tag{5}$$

The issue now is the choice of the three functionals. Before we proceed, we must define two types of functional we shall use, *invariant and sensitive to displacement* functionals. For simplicity, we shall call them simply "invariant" and "sensitive" functionals.

A functional Ξ of a function $\xi(x)$ is *invariant* if

$$\Xi(\xi(x + b)) = \Xi(\xi(x)), \quad \forall b \in \Re \qquad (I_1)$$

An invariant functional may also have the following additional properties:

- Scaling the independent variable by a, scales the result by some factor, $\alpha(a)$:

$$\Xi(\xi(ax)) = \alpha(a)\Xi(\xi(x)), \quad \forall a > 0 \qquad (i_1)$$

- Scaling the function by c, scales the result by some factor, $\gamma(c)$:

$$\Xi(c\xi(x)) = \gamma(c)\Xi(\xi(x)), \quad \forall c > 0 \qquad (i_2)$$

It can be shown (see Appendix A) that one can write:

$$\alpha(a) = a^{\kappa_\Xi}, \quad \text{and} \quad \gamma(c) = c^{\lambda_\Xi} \tag{6}$$

where parameters κ_Ξ and λ_Ξ characterise functional Ξ.

We shall also need to use functionals that have the following property: When applied on a periodic function u with period 2π, the result they yield is the same as the result we would get if the functional were to be applied to the original function minus its first harmonic $u^{(1)}$, denoted by $u^\perp \equiv u - u^{(1)}$:

$$Z(u) = Z(u^\perp) \qquad (si_1)$$

In table 1 we present some invariant functionals and their properties.

We may need to construct functionals with specific exponents κ and λ as defined by equations (6). For this purpose table 2 presents the rules with which we can construct extra functionals, from the knowledge of some appropriate functionals.

Invariant Functionals				
No	Functional	κ	λ	Properties
IF_1	$\int \xi(t)dt$	-1	1	I_1, i_1, i_2
IF_2	$(\int \lvert\xi(t)\rvert^q dt)^r$	-r	qr	I_1, i_1, i_2
IF_3	$\int \lvert\xi(t)'\rvert dt$	0	1	I_1, i_1, i_2
IF_4	$\int (t - SF_1)^2 \xi(t)dt$	-3	1	I_1, i_1, i_2
IF_5	$\sqrt{\frac{IF_4}{IF_1}}$	-2	0	I_1, i_1, i_2
IF_6	$max(\xi(t))$	0	1	I_1, i_1, i_2
IF_7	$IF_6 - min(\xi(t))$	0	1	I_1, i_1, i_2
IF_8	Amplitude of 1st harmonic of $\xi(t)$		1	I_1, i_2
IF_9	Amplitude of 2nd harmonic of $\xi(t)$		1	I_1, i_2, si_1
IF_{10}	Amplitude of 3rd harmonic of $\xi(t)$		1	I_1, i_2, si_1
IF_{11}	Amplitude of 4th harmonic of $\xi(t)$		1	I_1, i_2, si_1

Table 1: Some invariant functionals with their properties. For the definition of SF_1 see table 3.

A functional Z is called *sensitive* if

$$Z(\zeta(x+b)) = Z(\zeta(x)) - b, \quad \forall b \in \Re \qquad (S_1)$$

A sensitive functional of a periodic function is defined as follows: Let τ be the period of the function on which Z is defined. A function is called *τ-sensitive* if:

$$Z(\zeta(x+b)) = Z(\zeta(x)) - b_{(mod\ \tau)}, \quad \forall b \in \Re \qquad (S_2)$$

New Functionals			
No	Functional	κ	λ
1	$\Xi_1\Xi_2$	$\kappa_{\Xi_1}+\kappa_{\Xi_2}$	$\lambda_{\Xi_1}+\lambda_{\Xi_2}$
2	$(\Xi)^q$	$q\kappa_{\Xi}$	$q\lambda_{\Xi}$
3	Ξ_1/Ξ_2	$\kappa_{\Xi_1}-\kappa_{\Xi_2}$	$\lambda_{\Xi_1}-\lambda_{\Xi_2}$

Table 2: How to create new functionals.

.

A sensitive functional may also have the following properties:

- Scaling the independent variable, scales the result inversely:

$$Z(\zeta(ax)) = \frac{1}{a}Z(\zeta(x)), \quad \forall a > 0 \qquad (s_1)$$

Combining this with (S_1), we obtain (see Appendix B):

$$Z(\zeta(a(x+b))) = \frac{1}{a}Z(\zeta(x)) - b \qquad (s_{11})$$

and

$$Z(\zeta(ax+b)) = \frac{1}{a}Z(\zeta(x)) - \frac{b}{a} \qquad (s_{12})$$

- Scaling the function does not change the result:

$$Z(c\zeta(x)) = Z(\zeta(x)), \quad \forall c > 0 \qquad (s_2)$$

Table 3 lists some sensitive functionals and their properties.

3 Invariant feature construction

3.1 Theory

Suppose that the functionals we choose T, P and Φ are invariant, with T obeying property (i_1), P obeying properties (i_1) and (i_2) and Φ obeying property (i_2). We shall see how the value of the computed triple feature will be affected by the linear distortion of the image.

We start by observing that the triple feature of the distorted image will be given by:

$$\Pi(F, C_2) = \Phi\left(P\left(T\left(F(C_1; \phi_{old}, p_{old}, t_{old})\right)\right)\right) \qquad (7)$$

Sensitive Functionals				
No	Functional	κ	λ	Properties
SF_1	$\frac{\int t\xi(t)dt}{IF_1}$	-1	0	S_1, s_1, s_2
SF_2	x^* so that $\int_{-\infty}^{x^*} \xi(t)dt = \int_{x^*}^{+\infty} \xi(t)dt$	-1	0	S_1, s_1, s_2
SF_3	x^* so that $\int_{-\infty}^{x^*} \lvert\xi(t)'\rvert dt = \int_{x^*}^{+\infty} \lvert\xi(t)'\rvert dt$	-1	0	S_1, s_1, s_2
SF_4	Phase of 1st harmonic of $\xi(t)$		n/a	S_1 mod 2π,s_2
SF_5	Phase of 2nd harmonic of $\xi(t)$		n/a	S_1 mod π,si_1
SF_6	Phase of 3rd harmonic of $\xi(t)$		n/a	S_1 mod $2\pi/3$,si_1
SF_7	Phase of 4th harmonic of $\xi(t)$		n/a	S_1 mod $2\pi/4$,si_1

Table 3: Some sensitive functionals with their properties.

If we substitute from equations 1, 2 and 3, we obtain:

$$\Pi(F, C_2) = \Phi\left(P\left(T\left(F\left(C_1; \phi - \theta, \nu\left[p - s_0 \cos(\psi_0 - \phi)\right], \nu\left[t - s_0 \sin(\psi_0 - \phi)\right]\right)\right)\right)\right) \tag{8}$$

Due to property (i_1) and the invariance of T, this can be written as:

$$\Pi(F, C_2) = \Phi\left(P\left(\alpha_T\left(\nu\right) T\left(F\left(C_1; \phi - \theta, \nu\left[p - s_0 \cos(\psi_0 - \phi)\right], t\right)\right)\right)\right) \tag{9}$$

Due to property (i_2) obeyed by P, this is:

$$\Pi(F, C_2) = \Phi\left(\gamma_P\left(\alpha_T\left(\nu\right)\right) P\left(T\left(F(C_1; \phi - \theta, \nu\left[p - s_0 \cos(\psi_0 - \phi)\right], t\right)\right)\right) \tag{10}$$

Due to property (i_1) obeyed by P and its invariance, we have:

$$\Pi(F, C_2) = \Phi\left(\gamma_P\left(\alpha_T(\nu)\right) \alpha_P(\nu) P\left(T\left(F\left(C_1; \phi - \theta, p, t\right)\right)\right)\right) \tag{11}$$

Finally, if Φ obeys property (i_2) and is invariant, we have:

$$\Pi(F, C_2) = \gamma_\Phi\left(\gamma_P\left(\alpha_T\left(\nu\right) \alpha_P(\nu)\right)\right) \Phi\left(P\left(T\left(F\left(C_1; \phi, p, t\right)\right)\right)\right) \tag{12}$$

We can express this condition in terms of the exponents κ and λ of the functionals, to obtain:

$$\Pi(F, C_2) = \nu^{\lambda_\Phi(\kappa_T \lambda_P + \kappa_P)} \Pi(F, C_1) \tag{13}$$

Then an obvious condition for invariance is:

$$\lambda_\Phi(\kappa_T \lambda_P + \kappa_P) = 0 \tag{14}$$

If there is no scale difference between the objects to be matched, (ie if $\nu = 1$) then this condition is not necessary and any invariant functionals can be used, as long as they obey the necessary properties.

Now, let us assume that we choose functionals T and Φ to be invariant and functional P to be sensitive and obey property (s_{11}). Φ further obeys property si_1. Then equation 10 no longer follows from equation 9. Instead, applying property (s_{11}) of P we have:

$$\Pi(F, C_2) = \Phi\left(\frac{1}{\nu} P\left(T\left(F\left(C_1; \phi, p, t\right)\right)\right) + s_0 \cos(\psi_0 - \phi)\right) \tag{15}$$

Finally, due to si_1 property of Φ, we obtain:

$$\Pi(F, C_2) = \gamma_\Phi\left(\frac{1}{\nu}\right)\Phi\left(P\left(T\left(F\left(C_1; \phi, p, t\right)\right)\right)\right) \tag{16}$$

or equivalently:

$$\Pi(F, C_2) = \nu^{-\lambda_\Phi}\Pi(F, C_1) \tag{17}$$

We can see that if we choose Φ so that

$$\lambda_\Phi = 0, \tag{18}$$

the calculated triple feature will again be invariant to rotation, translation and scaling.

However, conditions (14) and (18) are too restrictive. One can generalise the relationships between the triple features computed in the two cases as

$$\Pi(F, C_2) = \nu^{-\omega}\Pi(F, C_1) \tag{19}$$

where $\omega \equiv -\lambda_\Phi(\kappa_T\lambda_P + \kappa_P)$ for equation (13) and $\omega \equiv \lambda_\Phi$ for equation (17). Since we choose the functionals we use, ω is known. Then every triple feature we compute may be normalised

$$\Pi_{norm}(F, C_1) = \sqrt[\omega]{|\Pi(F, C_1)|}\, sign(\Pi(F, C_1)) \tag{20}$$

and relationship (19) may be simplified to:

$$\Pi_{norm}(F, C_2) = \nu^{-1}\Pi_{norm}(F, C_1). \tag{21}$$

The ratio of a pair of such features becomes an invariant. Table 4 shows the combinations of functionals used to produce invariant triple features for the example images shown in figure 3.2. Each image is depicted in 4 different versions, shifted, rotated and translated with respect to each other. The triple features computed for each fish are pretty stable over the four versions of the same fish, and different from some of the features computed for the other fish. Note that the highest inaccuracies refer to image B3 which is the smallest version of fish B. Each image is 200×400 pixels. The trace transform is computed by sampling parameter p with 200 values in the range $[0, \sqrt{200^2 + 400^2}]$, and parameter ϕ with 240 values in the range $[0, 360^o]$. The trace functional was calculated by sampling each trace line with equidistant points with step $\frac{\sqrt{200^2+400^2}}{200}$. These parameters are kept constant for all calculations. The stability of the computed triple features over the various versions of the same image improves as the sampling of these parameters becomes denser, at the expense of computational efficiency. In figure 4 we plot pairs of these ratio features against each other to make clearer the stability some of them exhibit over the different versions of the same object. The features presented in table 4 are only very few examples of the total number of features that could be computed.

To add robustness to the process, instead of just two, a whole set of normalised features may be computed: Π_{norm_i} for $i = 1, \ldots, N$. They will constitute a feature vector, the norm of which is linearly proportional to the scaling coefficient ν. The direction of this vector in feature space is invariant to rotation, translation and scaling. Then the individual invariant features that can be used are the direction coefficients of this feature vector in feature space, defined as:

$$\Pi_{NORM\_i}(F, C_1) = \frac{\Pi_{norm\_i}}{\sqrt{\sum_{i=1}^{N} \left(\Pi_{norm\_i}(F, C_1)\right)^2}} \tag{22}$$

We shall call $\Pi_{NORM\_i}(F, C_1)$ a *normalised triple feature.*

Note that the invariant features produced that way are not necessarily independent. A feature selection algorithm applied at the end would be able to select the best independent features that are necessary for any classification problem. However, feature selection is a whole topic on its own and it is beyond the scope of this paper.

3.2 An application to image database retrieval

Figure 5 shows all the entries of a database consisting of 94 objects. Each object in the database was made the subject of a query after being rotated, scaled and translated in some random way. Figure 6 shows some example queries and the corresponding best five results of the database search. The similarity measure used to identify the answer was defined as follows: The features were scaled so that all varied within the same range. The absolute differences between the scaled features of the query and each of the entries of the database were ranked, and the average of the smallest two thirds of them was calculated as the similarity measure. Other measures of similarity were also tried, like the median of the absolute differences between the scaled features of the query and each of the entries of the database. The results obtained by them were similar with those obtained by the average of the two thirds most similar features. The numbers next to each reply image in figure 6 are the measures of similarity computed. For comparison, we also present in figure 7 the best five responses when the same query was posed to the same database using the system developed by Abbasi et al [1, 2]. This system is purely shape based, using a scale space representation of each object extracted from its outlining contour.

To check the robustness of our approach, and identify its breaking points, we performed many systematic experiments using different scaling factors and different levels and types of added noise. Table 5 shows the results of the experiments when the robustness to scaling is examined. The algorithm performs reasonably well, at scales down to 0.4, in the sense that in 90 or 91 out of the 94 times the correct object was included among the best five returned answers. To give an idea about the severity of the performed experiment, we show in figure 8 the outcome of a successful and an unsuccessful query when the scaling factor was 0.1.

To check robustness to noise we considered two cases: In the first case we assume that the noise has only corrupted the object, but not the background. This is the case when the grey values of the object have been distorted, but somehow the object has been extracted from the background of the image. Table 6 shows the results of adding Gaussian noise to the object with zero mean and standard deviation varying from 20 to 300. Note that the maximum grey value is 255, and the very high levels of noise are not purely Gaussian as the values are clipped. It may be considered as subgaussian noise with saturation. Figure 9 shows an example of an object and its noisy version with added Gaussian noise with $\sigma = 50, 100, 200$ and 300. Underneath each object we show its trace transform obtained with the third functional from table 1. We can see that even at very high levels of noise the trace transform does not appear to have changed noticeably.

Table 7 shows the results of experimenting with query objects that have been shifted and rotated by random amounts, and scaled as well as corrupted by additive Gaussian noise. The entries of the table show how many times the query object was the first one retrieved by the database search. We used only those levels of noise which in the case of no rotation, scale and translation could retrieve all 94 objects as the first choice.

The results of tables 5, 6 and 7 show that the system is very robust to a number of different distortions. However, this is not the case if the noise is added to the whole image. Figure 10 shows an example of an image and some noisy versions of it when the Gaussian noise added to the whole image has standard deviation $\sigma = 4, 8, 10$ and 15. (Grey values outside the range 0–255 are clipped.) The trace functional used this time is the median of the values along the tracing line. Even at the relatively moderate levels of noise used here, the trace transform has been noticeably distorted. Table 8 shows the results of experiments when the effects of noise to the whole image are combined with random rotation and translation, and with specific amounts of scaling. The catastrophic deterioration in performance can be explained by the fact that each object occupies only a very small area in the image and noise dominates what is computed along each tracing line. This effect could be reduced if instead of the whole image the minimum enclosing rectangle of each object is considered. By not doing that we actually put our system under very extreme functioning conditions. When the noise affects only the object grey values, the system degrades gracefully because it manages to pick up shape information. When the noise affects the whole image, shape information as well as grey value information is destroyed.

Tables 9 and 10 show the results of a similar experiment where the noise used is salt and pepper. The numbers represent the percentages of pixels that had their values altered: half of them were given value 0 and half value 255.

Note that the columns in the various tables of results we produce which correspond to zero levels of noise are not identical. This is because for each series of experiments we performed, we rotated and shifted each object by random amounts, generated using a random number generator. So, if one compares the results of the no-noise columns of tables 7–10, one can get an idea how the system performs for different (arbitrary) amounts of rotation and translation.

4 Linear transformation parameter estimation

4.1 Theory

In many problems of industrial inspection, or identity verification, one has to compare two patterns with each other in order to identify faults or discrepancies. In these cases, the tested pattern is not necessarily in complete registration with the reference pattern. For example, in the problem of

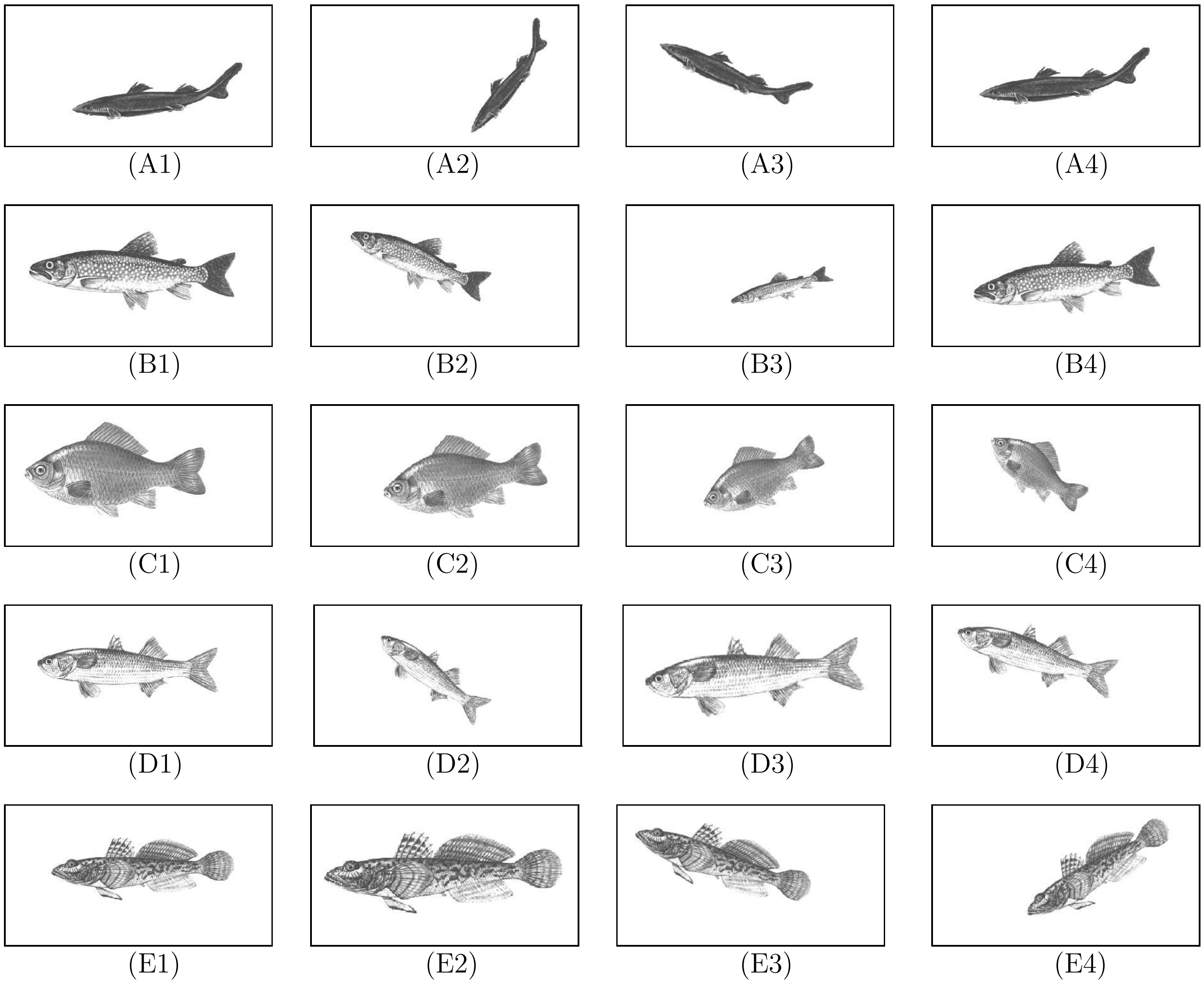


Figure 3: Each line represents four different versions of the same fish.

Fish	**Ratio of Triple Features** $\Pi_{norm\_1}/\Pi_{norm\_2}$			
	version 1	version 2	version 3	version 4
	$\Pi_1 \equiv \frac{\Pi_{norm\_1}(T=IF_5,P=IF_1,\Phi=IF_{11})}{\Pi_{norm\_2}(T=IF_5,P=IF_1,\Phi=IF_9)}$			
A	0.865	0.864	0.864	0.864
B	0.819	0.821	0.856	0.821
C	0.711	0.712	0.711	0.710
D	0.822	0.822	0.821	0.820
E	0.796	0.796	0.796	0.797
	$\Pi_2 \equiv \frac{\Pi_{norm\_1}(T=IF_5,P=IF_3,\Phi=IF_3)}{\Pi_{norm\_2}(T=IF_5,P=SF_2,\Phi=IF_{10})}$			
A	38.385	40.409	35.467	37.792
B	111.991	98.949	133.929	95.138
C	50.840	44.792	43.095	43.240
D	57.544	45.640	52.430	50.378
E	94.678	96.242	90.132	86.460
	$\Pi_3 \equiv \frac{\Pi_{norm\_1}(T=IF_4,P=IF_4,\Phi=IF_7)}{\Pi_{norm\_2}(T=IF_4,P=IF_4,\Phi=IF_6)}$			
A	0.933	0.932	0.932	0.932
B	0.893	0.895	0.894	0.892
C	0.881	0.881	0.883	0.883
D	0.906	0.905	0.903	0.906
E	0.838	0.840	0.837	0.835
	$\Pi_4 \equiv \frac{\Pi_{norm\_1}(T=IF_4,P=IF_4,\Phi=IF_{11})}{\Pi_{norm\_2}(T=IF_4,P=IF_4,\Phi=IF_9)}$			
A	0.694	0.685	0.683	0.686
B	0.914	0.918	1.007	0.916
C	0.839	0.835	0.833	0.842
D	1.064	1.064	1.066	1.061
E	0.634	0.642	0.634	0.634
	$\Pi_5 \equiv \frac{\Pi_{norm\_1}(T=IF_2,P=IF_1,\Phi=IF_{11})}{\Pi_{norm\_2}:(T=IF_2,P=IF_1,\Phi=IF_9)}$			
A	0.391	0.393	0.392	0.391
B	0.408	0.410	0.422	0.409
C	0.274	0.278	0.275	0.272
D	0.398	0.397	0.401	0.401
E	0.339	0.338	0.343	0.342

Table 4: Example triple features computed for the fish images

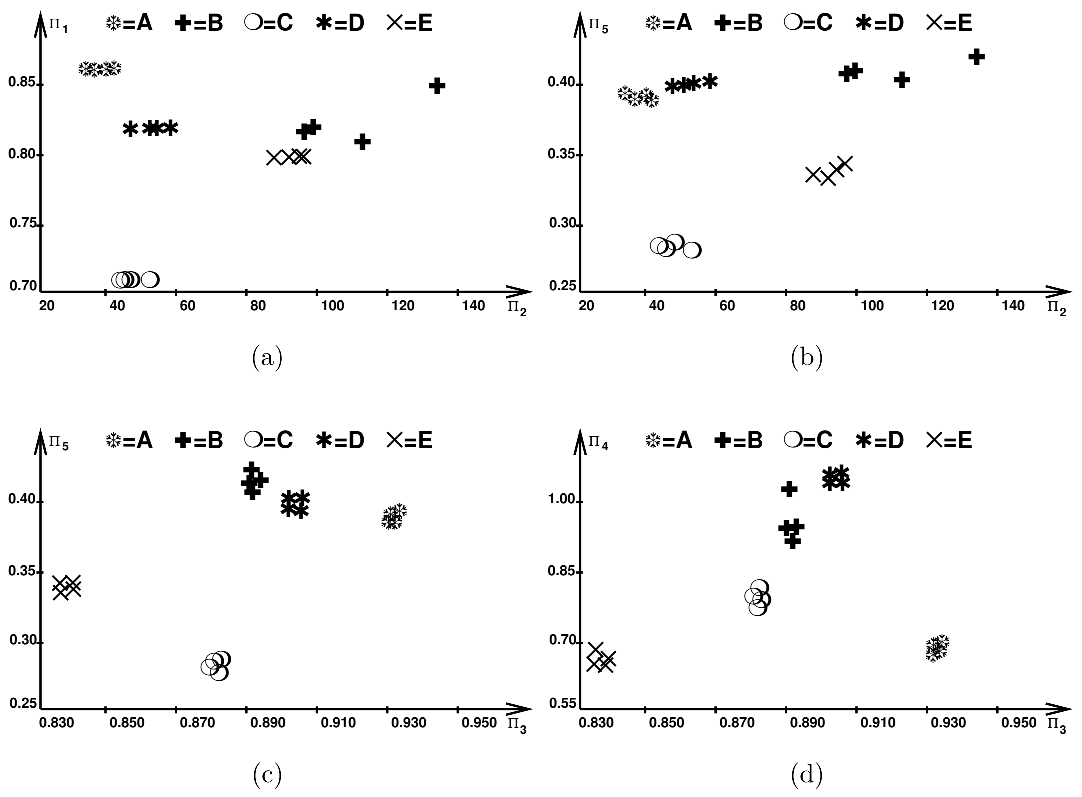


Figure 4: Plotting pairs of invariant features for the images of figure 3.2.

ceramic tile inspection, several tiles have patterns printed on them. To check for faults, the obvious route is to subtract the reference pattern from the test pattern. However, the tiles, as they come out from the production line, are placed on the conveyor belt in an almost random way. A camera at the zenith of the conveyor belt captures the tile image as it passes underneath. This image has to be compared with the reference image in the memory of the computer. Before the two patterns are subtracted, the captured test image must be registered with the coordinate system of the reference image. This involves the identification of the rotation and translation parameters between the two patterns, the subsequent registration of the two images, and finally the comparison for the detection of faults.

The usual way of identifying the parameters of rotation and translation is the phase correlation method [9, 3, 5]: This involves using the Fourier transform of the images. The Fourier transforms of two images that only differ by a translation, are identical, apart from a complex exponential factor multiplying the Fourier transform of one of the images, with phase equal to the translation parameter. Therefore, the inverse Fourier transform of the ratio of the two Fourier transforms is simply an impulse

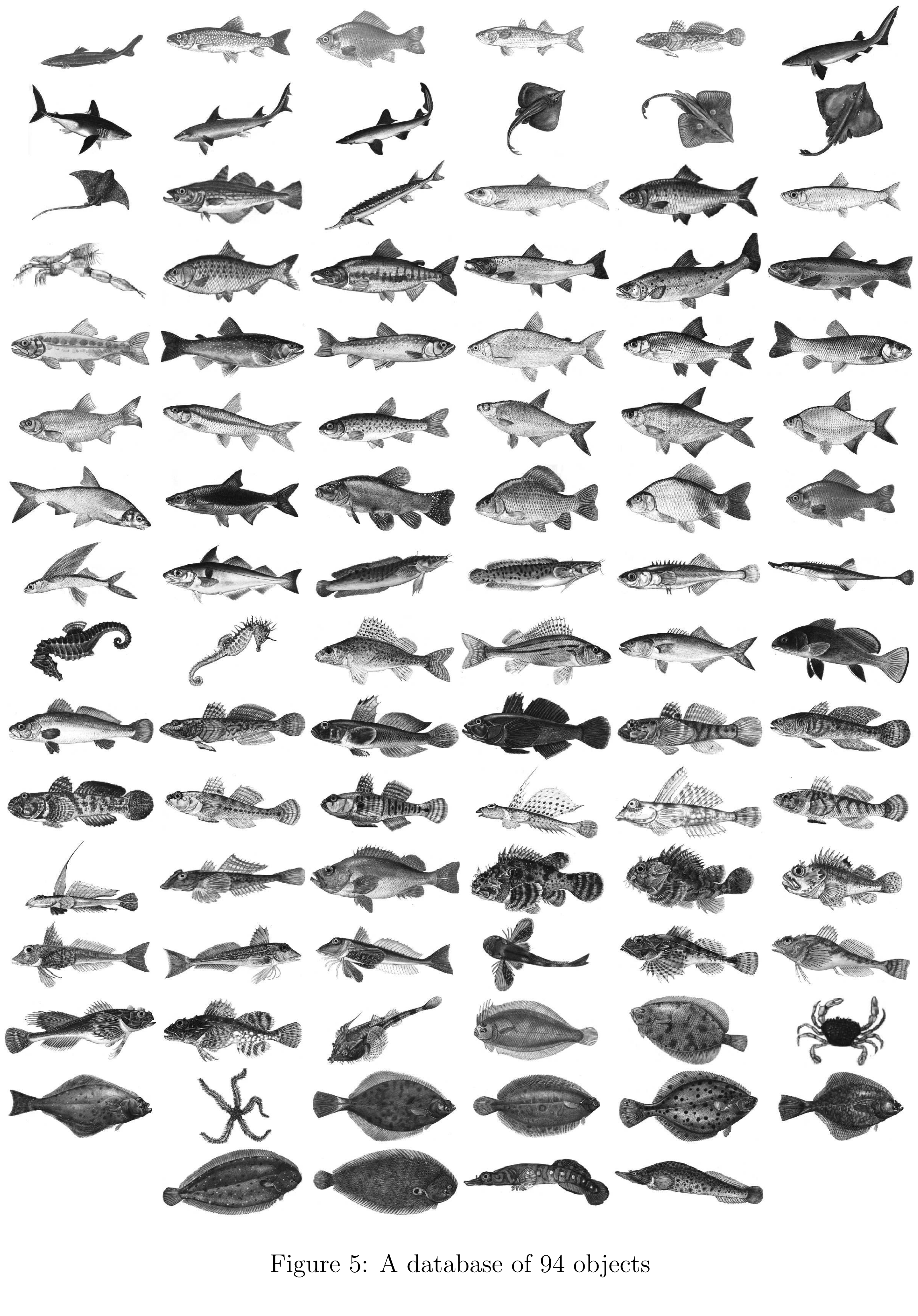


Figure 5: A database of 94 objects

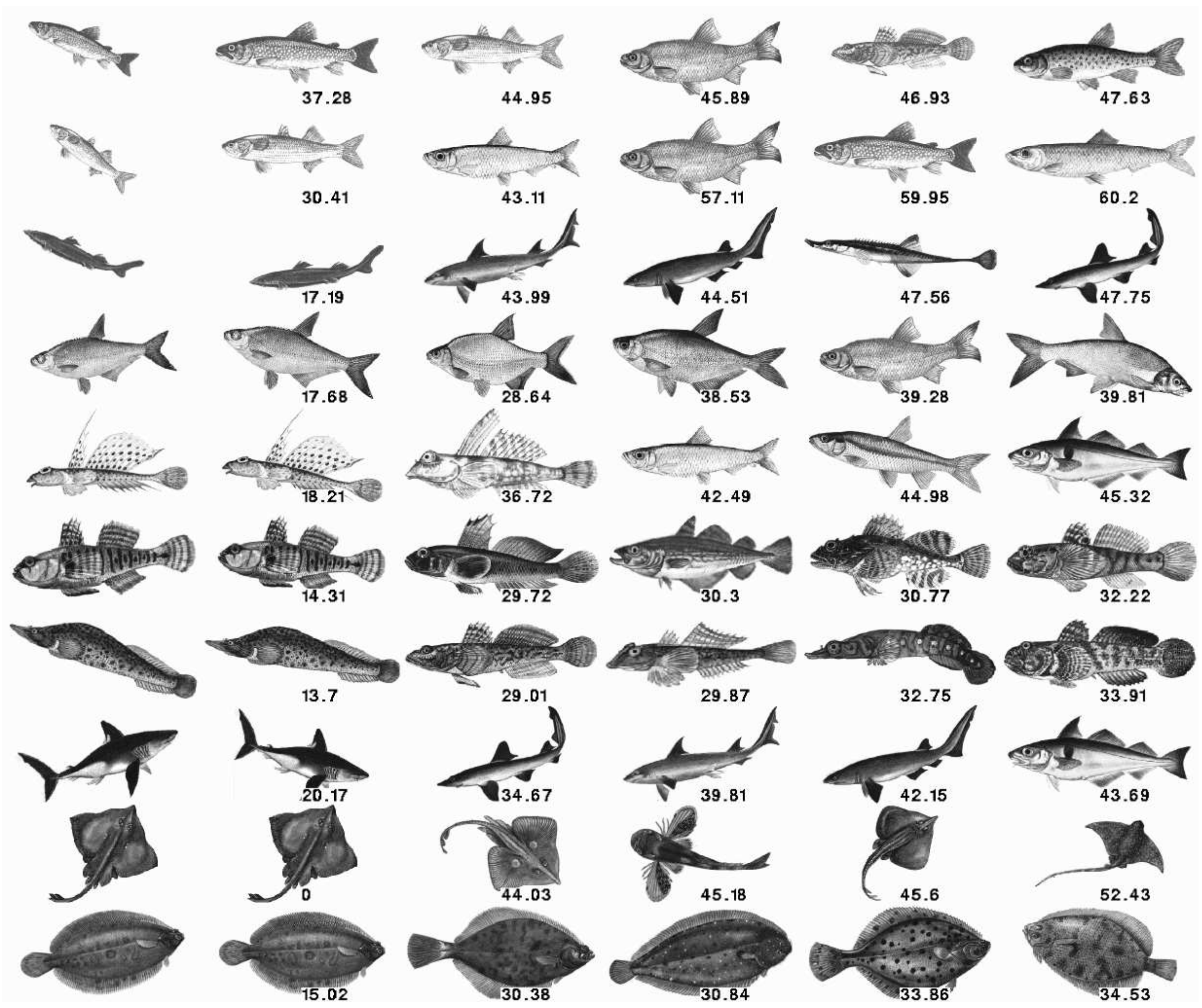


Figure 6: Querying the database with the images in the first column, produces the answers in all other columns arranged in order of similarity. The values of the similarity measure are given next to each object retrieved.

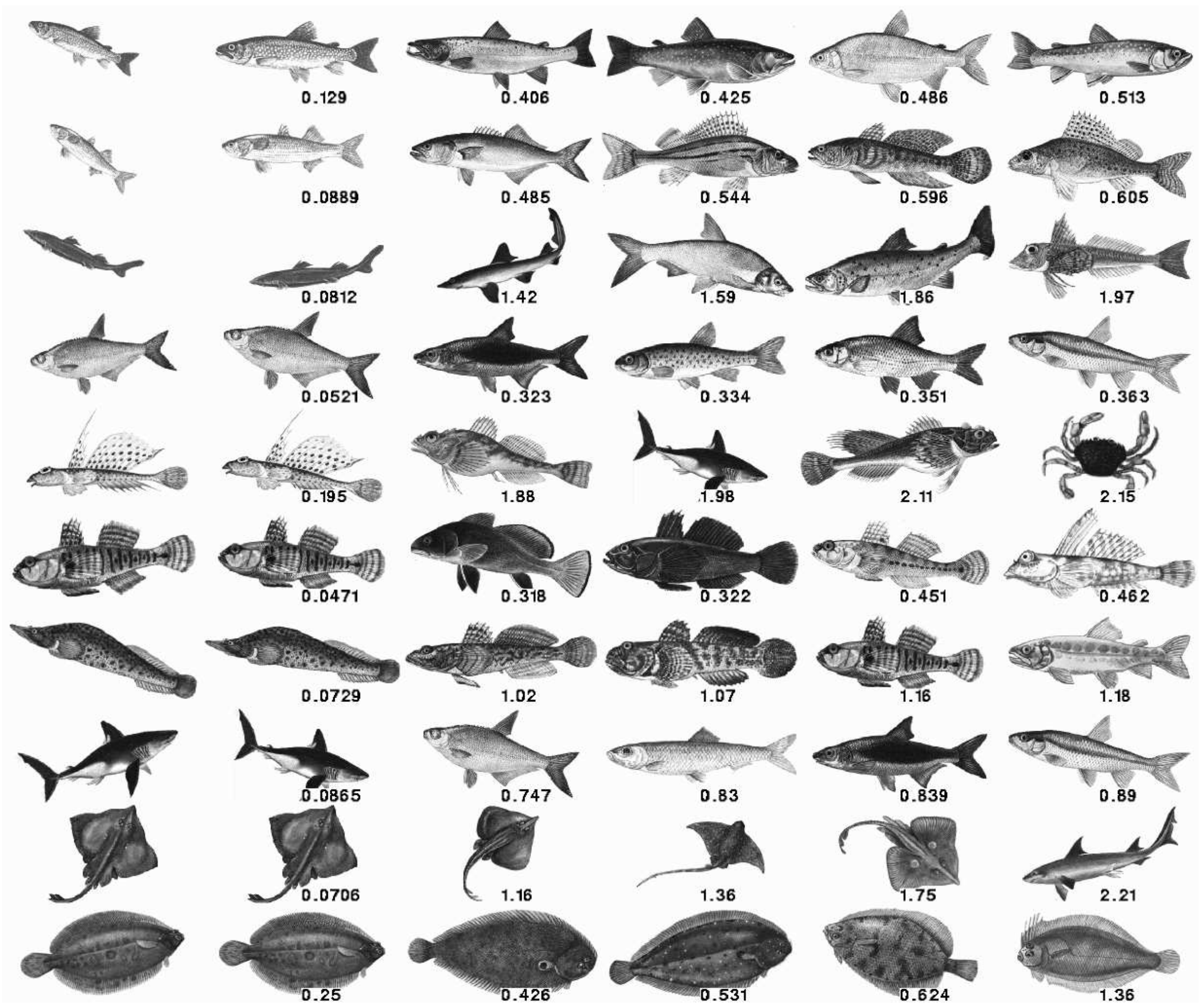


Figure 7: Comparison with a shape based image retrieval method described in [1, 2]: Querying the database with the images in the first column, produces the answers in all other columns arranged in order of similarity. The values of the similarity measure are given next to each object retrieved. These results were obtained by S Abbasi.

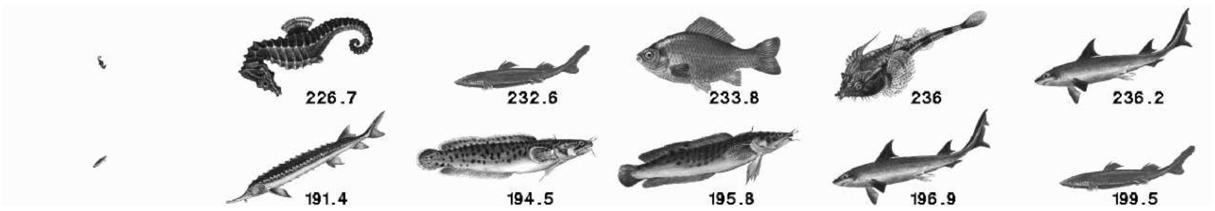


Figure 8: Querying the database with the images in the first column, produces the answers in all other columns arranged in order of similarity. The values of the similarity measure are given next to each image retrieved. The query object is 1/10 the size of the object in the database, rotated and translated. The query objects are about 30 pixels long. In the top row, the correct image was returned as the first response to the query. In the bottom row, the correct response was the third one.

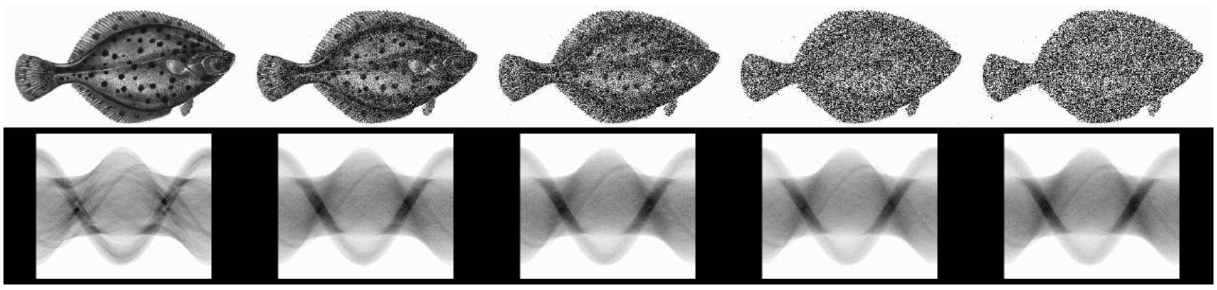


Figure 9: An original object and its versions distorted by additive Gaussian noise of standard deviation 50, 100, 200 and 300, respectively from left to right. (The noise was added only inside the object). At the bottom row the corresponding trace transforms computed using the third functional in table 1.

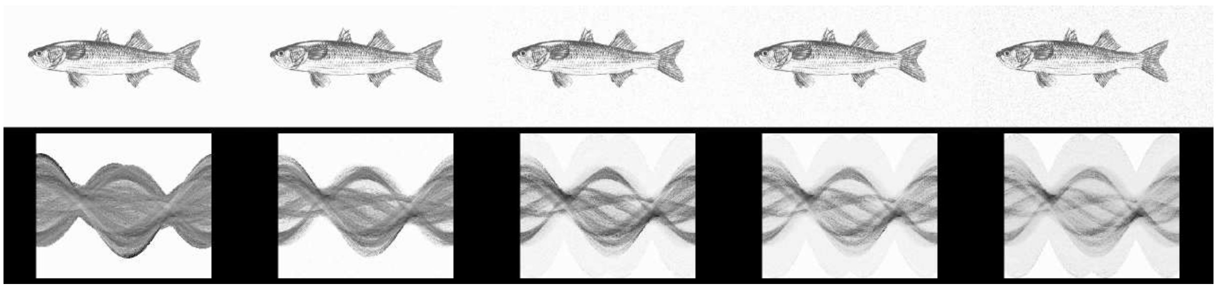


Figure 10: An original image and its versions distorted by additive Gaussian noise of standard deviation 4, 8, 10 and 15, respectively from left to right. (The noise was added to the whole image). At the bottom row the corresponding trace transforms computed using the median weighted by the absolute values of the pixels as the trace functional.

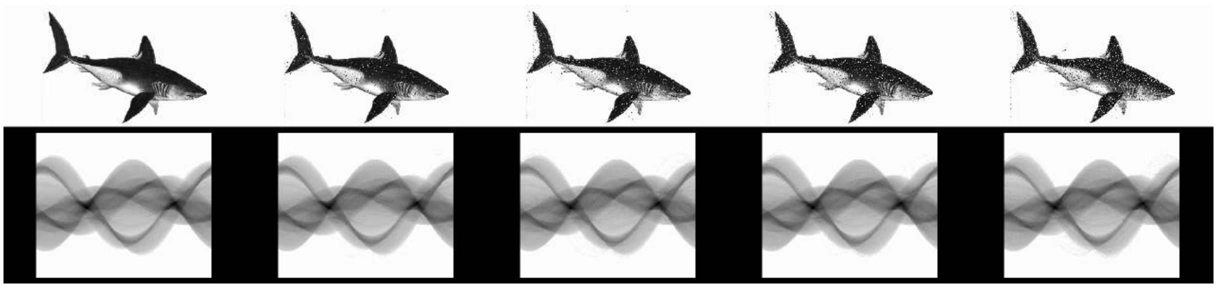


Figure 11: An original object and its versions distorted by salt and pepper noise. The percentage of altered pixels is 4%, 8%, 10% and 15%, respectively from left to right (half of these pixels were set to 0 and half to 255). (The noise was added only inside the object). At the bottom row the corresponding trace transforms computed using the second functional in table 1 with $p = 2$ and $r = 0.5$.

Scale factor	1st	2nd	3rd	4th	5th	total
0.9	94					94
0.8	94					94
0.7	91	3				94
0.6	83	8		1	2	94
0.5	81	7	1	1		90
0.4	70	11	8	2		91
0.3	62	16	3	3	1	85
0.2	47	17	11	7	4	86
0.1	18	10	5	9	10	52

Table 5: Robustness to scale: The first column shows the scaling factor with which each object of the database was reduced in size in order to make it a query object. The other columns show how many objects out of the 94 in the database were retrieved as the 1st, 2nd, etc choice when they were used a the query object. The last column gives the total number of objects that were included in the first five choices (sum of the previous columns).

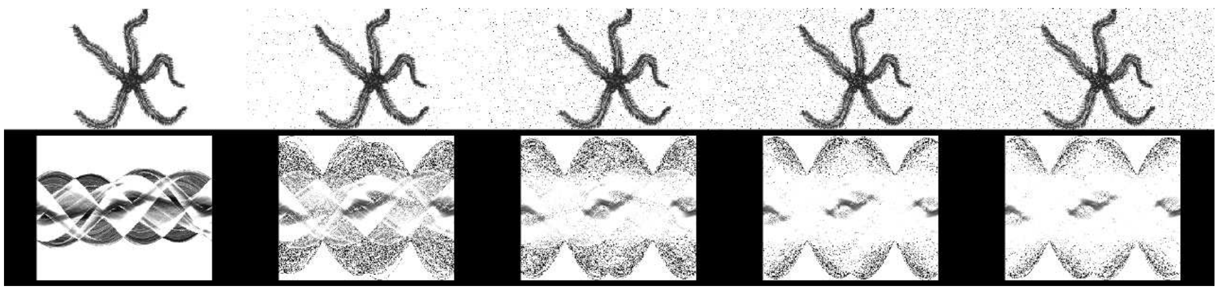


Figure 12: An original image and its versions distorted by salt and pepper noise. The percentage of altered pixels is 1%, 3%, 5% and 6%, respectively from left to right (half of these pixels were set to 0 and half to 255). (The noise was added to the whole image). At the bottom row the corresponding trace transforms computed using the median as the trace functional.

function, the coordinate position of which identifies the translation vector between the two images. Other variations of this approach have also been proposed [4].

In this section, we show how the Trace transform can be used for the same task.

The issue now is the choice of the three functionals. We start from equation 17 obtained for T and Φ invariant and P sensitive.

We can see that with the choice of the appropriate functionals we may be able to recover the scaling factor between two images. All we have to do is to calculate a certain triple feature for the two images and take the ratio of the two values. To add robustness to the process, several triple features of the appropriate type may be computed and several estimates of the same parameter made, with an average value extracted from them at the end.

Standard deviation	1st	2nd	3rd	4th	5th	total
20	94					94
30	94					94
40	90	2	2			94
50	88	3	1	2		94
60	86	5	1		1	92
70	82	2	5	1	1	91
80	79	4	3	3	2	91
100	70	10	2	1	3	86
120	60	13	4	2	3	82
140	59	8	8	3	1	79
160	55	11	7	2	4	79
180	47	14	12	4	1	78
200	49	14	10	5	2	80
220	45	11	8	6	4	74
240	47	11	5	7	6	76
260	41	18	10	4	3	76
280	42	11	8	5	6	72
300	38	13	9	8	6	74

Table 6: Robustness to additive Gaussian noise: The first column shows the standard deviation of the noise. The other columns show how many objects out of the 94 in the database were retrieved as the 1st, 2nd, etc choice when the noise was added to the object region only. The last column gives the total number of objects that were included in the first five choices (sum of the previous columns).

Suppose now that we wish to recover the rotation angle between two images.

Let us assume that we choose functional T to be invariant, P to be either invariant or sensitive, and functional Φ to be τ-sensitive, with $\tau \equiv \frac{2\pi}{n}$, and obeying property (s_2). Then it can be shown that

$$\Pi(F, C_2) = \Phi\left(P\left(T\left(F\left(C_1; \phi, p, t\right)\right)\right)\right) + \theta = \Pi(F, C_1) + \theta_{\text{mod } \tau} \tag{23}$$

We can see, therefore, that with the appropriate combination of functionals, we may be able to identify the rotation angle θ, by simply finding the difference between the values of a triple feature calculated for two versions of the same image that are scaled, shifted and rotated with respect to each other.

The translation parameters can be recovered as follows: Before we apply functional Φ, and after we have applied functionals T and P, we have a periodic function $h_2(\phi) = \frac{1}{\nu} P\left(T\left(F\left(C_1; \phi - \theta, p, t\right)\right)\right) + s_0 \cos(\psi_0 - \phi)$. To see that, we start by defining:

$$h_2(\phi) \equiv P\left(T\left(F(C_2; \phi, p, t)\right)\right) = P\left(T\left(F(C_1; \phi_{old}, p_{old}, t_{old})\right)\right) \tag{24}$$

Scale factor	σ of noise			
	0	10	20	30
1	94	94	94	94
0.9	94	94	94	93
0.8	94	94	92	87
0.7	91	89	88	79
0.6	85	85	83	77
0.5	80	82	73	72
0.4	66	70	69	61
0.3	59	61	59	55

Table 7: Performance of database retrieval when the object is scaled by the factor shown in the first column, rotated and translated in a random way, and its grey values have been distorted by additive Gaussian noise with standard deviation shown along the first row (noise was added to object only). The entries are the number of times the query object was the first found by the system (out of 94 objects in the database).

If we substitute from equations 1, 2 and 3 we obtain:

$$h_2(\phi) = P\left(T\left(F\left(C_1; \phi - \theta, \nu\left[p - s_0\cos(\psi_0 - \phi)\right], \nu\left[t - s_0\sin(\psi_0 - \phi)\right]\right)\right)\right) \tag{25}$$

Due to property (i_1) and the invariance of T this can be written as:

$$h_2(\phi) = P\left(\alpha_T(\nu)\, T\left(F\left(C_1; \phi - \theta, \nu\left[p - s_0\cos(\psi_0 - \phi)\right], t\right)\right)\right) \tag{26}$$

Due to property (s_{11}) obeyed by P, this is:

$$h_2(\phi) = \frac{1}{\nu} P\left(T\left(F\left(C_1; \phi - \theta, p, t\right)\right)\right) + s_0\cos(\psi_0 - \phi) \tag{27}$$

From the reference image we can compute the corresponding function $h_1(\phi) \equiv P(T(F(C_1; \phi, p, t)))$. Equation (27) then becomes:

$$h_2(\phi) = \frac{1}{\nu} h_1(\phi - \theta) + s_0\cos(\psi_0 - \phi) \tag{28}$$

Since we already know the values of ν and θ, we can find the value of $s_0\cos(\psi_0 - \phi)$. In practice, we need only compute the first harmonic of function $h_2(\phi) - \frac{1}{\nu}h_1(\phi - \theta)$. This first harmonic has the form $a\cos\phi + b\sin\phi$. Equating it with the last term on the right hand side of (28), we obtain:

$$a\cos\phi + b\sin\phi = s_0\cos(\psi_0 - \phi) \tag{29}$$

Note that by choosing $\phi = 0$ and $\phi = \pi/2$ we can easily identify the shifting vector $(s_0\cos\psi_0, s_0\sin\psi_0)$ with (a, b).

Scale factor	σ of noise						
	0	2	4	6	8	10	15
1	94	68	61	61	52	52	42
0.9	94	77	66	55	55	53	37
0.8	94	81	72	60	56	45	25
0.7	91	68	54	53	42	31	14
0.6	85	52	40	27	11	8	5
0.5	80	31	18	8	6	6	3
0.4	66	13	4	5	3	3	2
0.3	59	3	3	4	2	2	0

Table 8: Performance of database retrieval when the object is scaled by the factor shown in the first column, rotated and translated in a random way, and Gaussian noise with standard deviation shown along the first row has been added to the whole image. The entries are the number of times the query object was the first found by the system (out of 94 objects in the database).

We can easily verify that combinations (IF_2, IF_{2a}, SF_7), (IF_2, IF_6, SF_7), (IF_3, IF_{2a}, SF_7), (IF_3, IF_6, SF_7), (IF_4, IF_{2a}, SF_7), (IF_4, IF_6, SF_7), (IF_2, SF_1, SF_5), (IF_3, SF_1, SF_5) and (IF_4, SF_1, SF_5) for T, P and Φ respectively, satisfy relationship (23). Here IF_{2a} means IF_2 for $q = 2$ and $r = 0.5$.

4.2 An application to token verification

Figure 13a shows a reference image and 13b a rotated and translated version of it. When the transformation parameters were calculated and the two images were brought into registration and subtracted, the residual map was blank and that is why it is not reproduced here. Figures 13c to 13n show various faulty versions of the same image and the corresponding residual maps obtained.

The whole algorithm works with grey images, and although the above discussed images appear binary, no use of this fact is made in the calculation of the various functionals. Figure 13o is a grey reference image and 13p a rotated and translated version of it. No residual map is reproduced as it is blank after the two images are brought into registration and subtracted pixel by pixel. The remaining images in figure 13 are two faulty versions of the same grey image and the corresponding residual maps.

In all these experiments we used the same combinations of functionals, identified earlier.

To study the effects of noise we considered again two cases: the case when the depicted object itself has been only corrupted, and the case when the whole image has been corrupted. As before, the system shows its greatest sensitivity to noise when the noise is added to the whole image, not just the the object. Figure 14 shows some noisy objects and their difference computed after their rotation

Scale	% of noisy pixels						
factor	0%	2%	4%	6%	8%	10%	15%
1	94	94	93	90	88	86	80
0.9	94	93	88	86	84	83	74
0.8	94	90	86	81	79	73	63
0.7	91	83	79	79	68	58	57
0.6	86	74	71	62	62	59	48
0.5	74	67	60	57	56	53	45
0.4	71	61	60	53	48	42	38
0.3	61	55	52	47	45	37	34

Table 9: Performance of database retrieval when the object is scaled by the factor shown in the first column, rotated and translated in a random way, and its grey values have been distorted by salt and pepper noise. The numbers along the first row show the percentage of pixels having their grey values set either to 0 or to 255 (half and half). The entries are the number of times the query object was the first found by the system (out of 94 objects in the database).

and translation parameters with respect to the reference image in figure 13a have been identified. The type of noise used is salt and pepper noise. We can see that for the case of 40% noise the system begins to break down. Figure 15 shows noisy images of another version of the same object and their difference computed after their rotation parameters with respect to the reference image in figure 13a have been identified. Note that here the images are restricted to be the minimum enclosing rectangle of the object in order to decrease the sensitivity of the method to noise, but also because this corresponds to a realistic situation where a token, for example, is entered into a slot machine: one expects that, if the token is imaged, the field of view of the imaging device will be restricted just around the token. We can see from this figure that the system begins to break down at about 11% salt and pepper noise.

Similar experiments were performed with Gaussian noise. We do not show any figures, but the system could tolerate noise inside the object with σ up to about 100, while it was starting to break down at $\sigma = 120$. When the noise was added to the minimum enclosing rectangle, the system was starting to break down at noise levels with $\sigma = 60$, while it could function well at $\sigma = 40$.

5 Change Detection

In a monitoring problem often the change we are looking for manifests itself as a change in the texture of the object. Examples of such situations arise in remote sensing where the phenomenon we monitor is change of land use, in reconnaissance, where the phenomenon we are looking for is, for

Scale	% of noisy pixels						
factor	0%	1%	2%	3%	4%	5%	6%
1	93	66	46	33	19	8	8
0.9	94	51	26	11	8	4	3
0.8	94	27	9	5	3	4	3
0.7	92	18	6	2	2	2	2
0.6	82	3	3	3	1	0	1
0.5	78	1	1	1	0	0	1
0.4	68	0	1	2	2	2	1
0.3	58	0	1	1	1	1	1

Table 10: Performance of database retrieval when the object is scaled by the factor shown in the first column, rotated and translated in a random way, and the values of the whole image have been distorted by salt and pepper noise. The numbers along the first row show the percentage of pixels having their grey values set either to 0 or to 255 (half and half). The entries are the number of times the query object was the first found by the system (out of 94 objects in the database).

example, increased activity in an area, in medical images where various progressive diseases manifest themselves by changes in the texture of the tissue in question, etc.

We must therefore, extract texture features from images that can reflect the change from one image to the next in a sequence of images of the object, obtained under more or less the same conditions. Trace transform is ideal for that as it allows the calculation of thousands of features which can be used in a training stage of the process during which those that exhibit the right correlations with the phenomenon under study may be identified.

Our approach is demonstrated with the help of a collection of 20 images of the same car park where we are looking for increased activity, expressed by the number of cars parked in it. These images in order of increasing number of cars they contain are shown in figure 16.

Among the hundreds of features tried, some combinations that appeared useful were combinations formed by choosing T to be any of the IF_1–IF_6 or SF_1, P any one of IF_1–IF_6 or SF_1–SF_3, and Φ any one of IF_1–IF_3, IF_6, or SF_4–SF_7.

To be able to detect change we must assume that the images are processed in a random sequence, with no prior knowledge of the classes we expect to see. The first image is used as a reference image, and all subsequent images are compared with it and a single similarity measure is computed for each image. This similarity measure must reflect the amount of change that we can perceive in the image. The triple features that we adopted were chosen so that they have this property. However, in order to have a single similarity measure, we calculated from them a single number for each image, by simply scaling and averaging them. The scaling is necessary because each feature varies within a different range of real numbers. The scaling is done in such a way that the range of each feature is roughly between 0 and 10. It is impossible to predict that the feature of *all* subsequent images will be exactly in this range, as some unexpected changes may happen to the inspected scene. However, this is not important as we are looking for trends and it does not matter if the characterising numbers spill out of the pre-specified range. Figure 17a shows this average "feature" computed for the 20 available images, with the images sorted in ascending order of cars present in them (the same order with which the images are presented in figure 16). Figures 17b–17e show the absolute difference values that are obtained for each image if it is compared with the reference one, which is taken to be image 16b, 16f, 16p and 16t with 6, 9, 19 and 21 cars respectively.

It is worth stressing that the method presented here is not equivalent to counting cars in each image. It is more about characterising the texture present in each image. Indeed, car counting amounts to blob counting. We tried to solve the same problem by adapting the approach of Song et al [14], successfully used in granite tile inspection [11]. This approach consists of splitting the grey image into binary planes using either bit-slicing or quantisation into 8 grey levels after histogram equalisation, and creating a stack of binary planes, each one flagging the corresponding pixels. Morphological processing was applied to each binary plane, followed by connected component labelling to identify individual blobs. Statistics concerning the distribution of the area, elongatedness, and compactness of the blobs were computed and then used to characterise the images depicting acceptable situations. Hundreds of experiments were performed for various values of the parameters. They are all reported in [11]. This approach failed to solve the problem in a robust and reliable way. We do not include any results here due to lack of space.

6 Computational complexity

The computational complexity of the method depends on the following problem parameters:

N_T:	the number of trace functionals;
N_P:	the number of diametrical functionals;
N_Φ:	the number of circus functionals;
n_p:	the number of samples of parameter p;
n_ϕ:	the number of samples of parameter ϕ;
n_t:	the number of samples of parameter t along the diagonal of the image;
C_T:	the number of operations per sample for trace functionals;
C_P:	the number of operations per sample for diametrical functionals;
C_Φ:	the number of operations per sample for circus functionals.

To compute a trace transform one has to compute $n_p n_\phi$ numbers. Each such number is computed by applying the trace functional to the image along the line parameterised by parameter t. If the line we use is the longest possible (diagonal), then n_t samples are involved. In other cases the number of samples is smaller. Therefore, to compute the whole trace transform one has to consider fewer than $n_t n_p n_\phi$ samples of parameter t. A more accurate estimation makes the number of t-samples to be equal to $n_t n_p n_\phi N_x N_y/(N_x^2 + N_y^2)$ where $N_x \times N_y$ is the size of the image. This number therefore, does not exceed $n_t n_p n_\phi/2$.

For each t-sample one should perform about C_T operations, so then all N_T trace transforms can be computed with $C_T N_T n_t n_p n_\phi/2$ operations. Then applying N_P diametrical functionals to the computed N_T trace transforms of size $n_p \times n_\phi$, requires $C_P N_P N_T n_p n_\phi$ operations and results in $N_P N_T$ strings of numbers of size n_ϕ. The last step is the application of N_Φ circus functionals to these strings, which requires $C_\Phi N_\Phi N_P N_T n_\phi$ operations and results in $N_\Phi N_P N_T$ triple features. Therefore, the overall number of operations needed to compute all $N_\Phi N_P N_T$ triple features is

$$C_T N_T n_t n_p n_\phi/2 + C_P N_P N_T n_p n_\phi + C_\Phi N_\Phi N_P N_T n_\phi$$

Typical numbers are $n_t = n_p = n_\phi = 10^2$, $N_T = N_P = N_\Phi = 10$. This results in approximately 10^7 operations to compute 10^3 triple features of the image. From this example one may see that the first term in the above expression dominates the result completely. This means that the most care should be given to the choice of the set of trace functionals.

As an example, we can estimate the number of operations needed for the experiment presented for image database retrieval. Each pattern in the database is characterised by approximately 10^3 triple features. If a new image comes along, it takes approximately 10^7 operations to build 10^3 features. Then comparison of this new image with 10^2 elements of the database takes $10^3 10^2 = 10^5$ operations.

7 Conclusions

We have introduced here a new tool for image processing, namely the Trace transform which is a generalisation of the Radon transform. We demonstrated its usefulness to a variety of tasks, including invariant feature construction, image registration and change detection.

The calculation of invariant features can be used to identify objects in an inventory type database. The Trace transform is a global one, so it cannot cope with occlusion or multiplicity of objects. Another limitation is that the features it constructs are not invariant to strong changes in illumination.

The second application we discussed makes use of the properties of the Trace transform to identify the rotation and translation parameters between two images. The scaling parameter can also be identified as a bi-product of the whole process. For completeness, we presented the whole theory, although we demonstrated with examples only the calculation of the rotation and translation parameters, that were most relevant to the problem of fraud detection. These parameters can be used to bring two images in complete registration, so that they can be subtracted to reveal even the most subtle differences between them. The accuracy with which they are computed depends on the density with which the continuous variables p, ϕ and t are sampled. Further, the method may rely on the calculation of a small or a large number of functionals, depending on the robustness required, and it is parallelizable to a large degree. This makes the method more powerful than the phase correlation method that is based on the Fourier transform. Because of the above properties, the method is ideal for identity verification for credit cards and other devices that need authentication on the basis of imprint images.

Using many lines to scan an image consumes a lot of computation time. However, by performing some pre-computation we can reduce the time to a reasonable level. Every line may be represented by a sequence of pixels. Let us suppose that we know beforehand the size of the images we are going to process and have chosen values for all quantisation parameters (number of lines, line quantisation, and ϕ and p sampling rates). Then we can pre-compute the pixel locations that make up each digitised tracing line and store them for further computations. Such a pre-computation reduces the computational time significantly. If several trace functionals are to be used, it saves time to compute them simultaneously along the precomputed tracing lines. Symmetry of trace functionals may also be taken into consideration in order to reduce computational time. For example, if a trace functional is symmetric (ie $T(f(t)) = T(f(-t))$, then we can compute the trace transform only for $0 \leq \phi < \pi$ and copy the result for $\pi \leq \phi < 2\pi$. The slowest example presented here was that of querying the database, which took about 4 sec per query. All other images in the other examples were processed in 0.2-1.0 sec, using a pentium PC of 300MHz. The calculation can be further accelerated, since the trace functional, for example, which is applied to all lines with which we trace an image, can be implemented in a parallel way. Similarly, the application of the diametric functional can be applied in parallel for all angles ϕ.

The most crucial part of our method is the development of different useful T- functionals using intuition and experience. Suppose that we have 10 options for functional T, 10 options for functional P and also 10 options for functional Φ (and note that they may coincide, that is, one T functional is allowed to be the same as one of the P- functionals). After these 30 computational procedures have been installed into the computer, we are able to produce $10 \times 10 \times 10 = 1000$ triple features. Definitely, most of them will not be useful; nevertheless we can investigate them and make the appropriate choice for the specific task with the help of experimentation. The Trace transform therefore, helps increase our options in search for effective characterisation of images. This point was demonstrated with the third application presented, concerned with change detection.

Acknowledgements: We are grateful to Ralf Jackson from DERA for supplying the images of figure 16, which are Crown Copyright and are used by permission. We are also greatful to Dr Mokhtarian who made available to us the fish database and to Dr Abbasi who run the experiment that produced figure 7. This work was supported by an EPSRC grant GR/M88600.

Appendix A

Let us assume that we scale the independent variable in property i_1 by $a = a_1 a_2$. We have:

$$\Xi\left(\xi(a_1 a_2 x)\right) = \alpha(a_1 a_2) \Xi\left(\xi(x)\right) \tag{30}$$

We may also write:

$$\begin{aligned} \Xi\left(\xi(a_1 a_2 x)\right) = \Xi\left(\xi(a_1(a_2 x))\right) &= \alpha(a_1) \Xi\left(\xi(a_2 x)\right) \\ &= \alpha(a_1)\alpha(a_2)\Xi\left(\xi(x)\right) \end{aligned} \tag{31}$$

By direct comparison of 30 and 31 we deduce that

$$\alpha(a_1 a_2) = \alpha(a_1)\alpha(a_2) \tag{32}$$

Let us introduce a function $Q(d) \equiv \ln[\alpha(e^d)]$ where $d \equiv \ln a$. We have:

$$\begin{aligned} Q(d_1 + d_2) = \ln[\alpha(e^{d_1+d_2})] &= \ln[\alpha(e^{d_1} e^{d_2})] \\ = \ln[\alpha(e^{d_1})\alpha(e^{d_2})] &= \ln[\alpha(e^{d_1})] + \ln[\alpha(e^{d_2})] \\ &= Q(d_1) + Q(d_2) \end{aligned}$$

Therefore, if α is a continuous function, Q is a linear function, ie it must have the form $Q(d) = \kappa d$ for some κ. So, $\ln[\alpha(e^d)] = \kappa d = \kappa \ln a = \ln a^\kappa$ and therefore $\alpha(a) = a^\kappa$.

Appendix B

Consider $Z(\zeta(a(x+b)))$. Let us define a new function $\eta(x) \equiv \zeta(ax)$. Then we can write:

$$Z(\zeta(a(x+b))) = Z(\eta(x+b)) \tag{33}$$

Because of property S_1, we may write:

$$Z(\eta(x+b)) = Z(\eta(x)) - b = Z(\zeta(ax)) - b \tag{34}$$

Making use of (s_1), we may write:

$$Z(\zeta(ax)) - b = \frac{1}{a} Z(\zeta(x)) - b \tag{35}$$

which proves that

$$Z(\zeta(a(x+b))) = \frac{1}{a} Z(\zeta(x)) - b \tag{36}$$

References

[1] S Abbasi, F Mokhtarian, and J Kittler, 1999. "Curvature Scale Space image in Shape Similarity Retrieval". Springer Journal of MultiMedia Systems, Vol 7, number 6, pp 467–476.

[2] S Abbasi, F Mokhtarian, and J Kittler, 2000. "Enhancing Curvature Scale Space Based Shape Retrieval for Objects with Shallow Concavities". Image and Vision Computing, Vol 18, pp 199–211.

[3] D Casasent, D Psaltis, 1976. "Position, rotation, and scale invariant optical correlation". Applied Optics, Vol 15, pp 1795–1799

[4] Q Chen, M Defrise and F Deconinck, 1994. "Symmetric Phase-only matched filtering of Fourier-Mellin Transforms for Image Registration and Recognition". IEEE PAMI, Vol 16, pp 1156–1168.

[5] C E Costa and M Petrou, 2000. “Automatic registration of ceramic tiles for the purpose of fault detection”. Machine Vision and Applications, Vol 11, pp 225–230.

[6] S R Deans, 1981. “Hough Transform from the Radon Transform”. IEEE PAMI, Vol 3, pp 185–188.

[7] S R Deans, 1983. “The Radon Transform and some of its applications”. Krieger Publishing Company.

[8] N G Fedotov, 1990. “Methods of stochastic geometry in pattern recognition”. Moscow, Radio and Communications, ISBN 5-256-00447-6.

[9] C D Kuglin, A F Blumenthal and J J Pearson, 1979. “Map-matching techniques for terminal guidance using Fourier phase information”. SPIE Vol 186 Digital Processing of Aerial Images, pp 21–29.

[10] A B J Novikoff, 1961. “Integral geometry as a tool in pattern perception: principles of self organisation”. Transactions of the University of Illinois Symposium on self-organisation, H von Foester and G Zopf, Jr, (eds), Pergamon Press, London, pp 347–368.

[11] M Petrou and P Garcia, 1998. “Statistical analysis of binary slices”, University of Surrey Technical report, No VSSP/1998-6.

[12] M Schael and H Burkhardt, 1998. “Error Detection in textures using invariant grey scale features”. Proceedings of the Workshop on Texture Analysis, Univ of Freiburg, Germany, Sept 1998.

[13] S Siggelkow and H Burkhardt, 1997. “Local invariant feature histograms for texture classification. Interner Bericht 3/97, Albert Ludwigs Universitat, Freiburg, Institut fur Informatik.

[14] K Y Song, J Kittler and M Petrou, 1996. Defect detection in random colour textures. Image and vision computing. Vol 14, pp 667–683.

[15] P Toft, 1996. “The Radon Transform: Theory and Implementation”. PhD thesis, Technical University of Denmark.

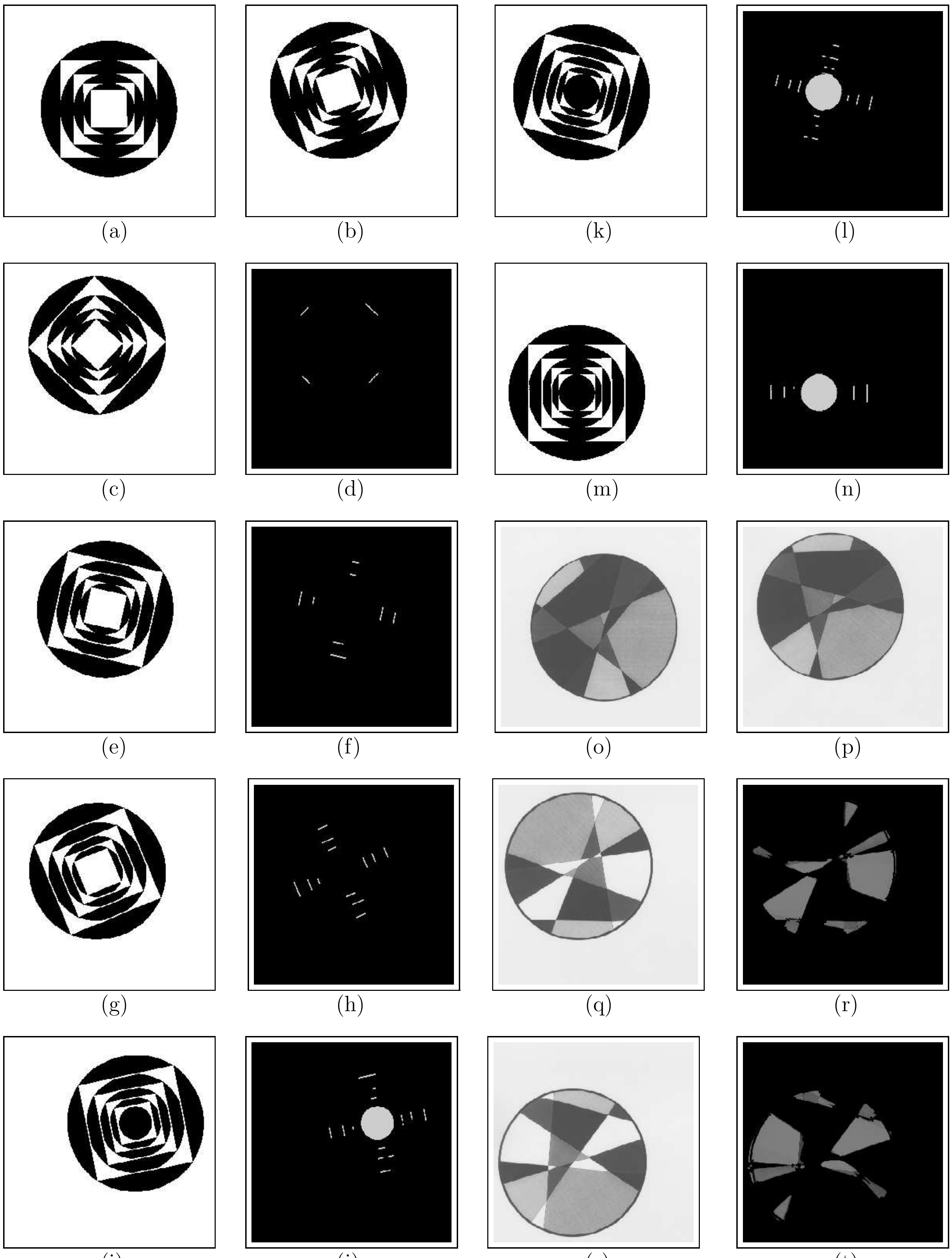


Figure 13: (a) & (o): reference images. (b) & (p): rotated and translated versions of them. The remaining images are pairs of faulty images and the results of subtracting them from the corresponding reference image.

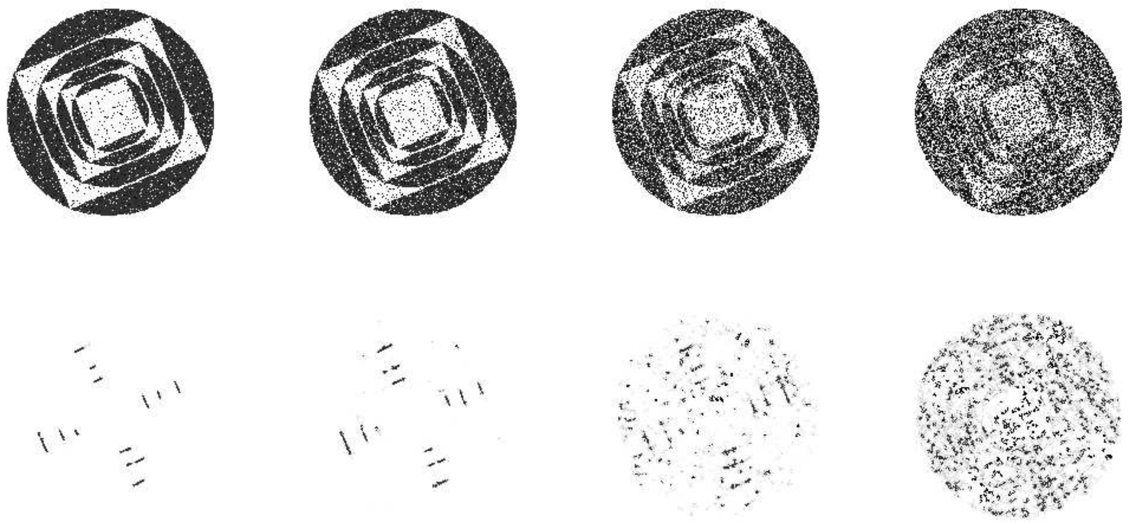


Figure 14: Noisy objects and their differences from the reference object of figure 13a extracted after their rotation and translation parameters with respect to that image have been identified. The levels of salt and pepper noise used from left to right are 10%, 20%, 40% and 60% respectively. The noise was added to the object region only.

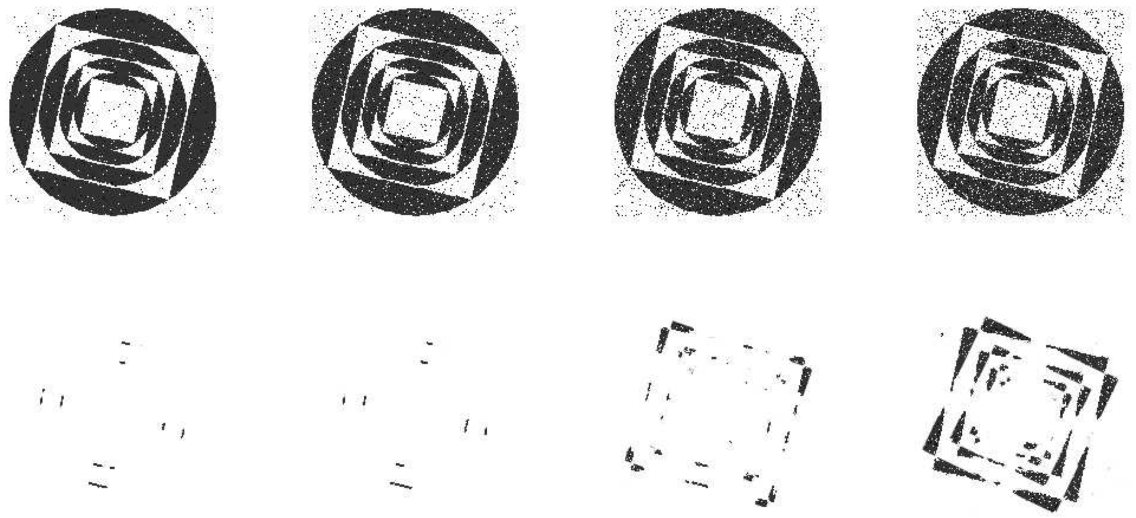


Figure 15: Noisy images and their differences from the reference image of figure 13a extracted after their rotation and translation parameters with respect to that image have been identified. The levels of salt and pepper noise used from left to right are 3%, 7%, 11% and 15% respectively. The noise was added to the minimum enclosing rectangle of the object.

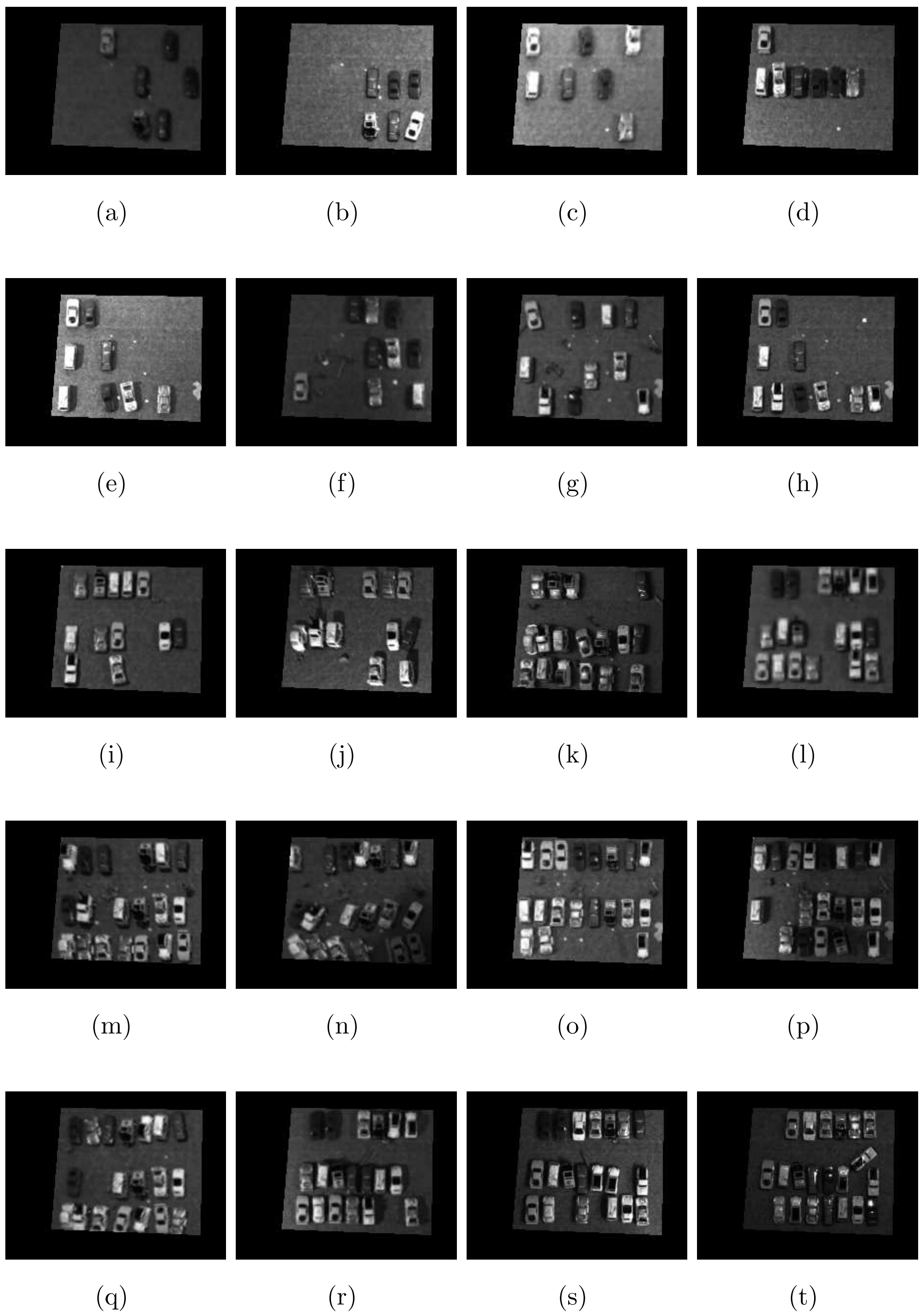
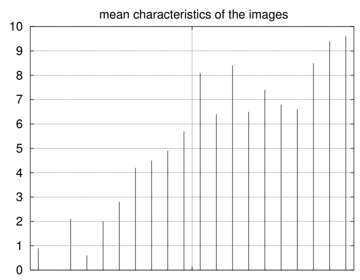
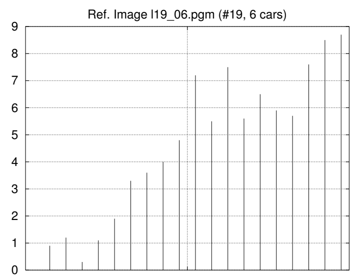


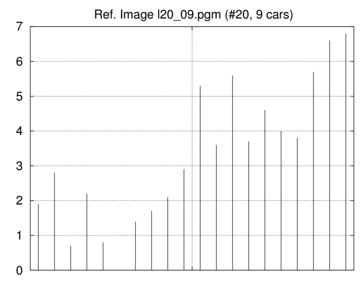
Figure 16: Images of a car park that have to be characterised in terms of the level of use of the car park.



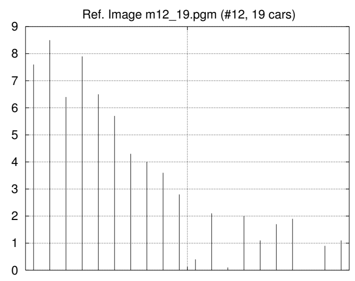
(a)



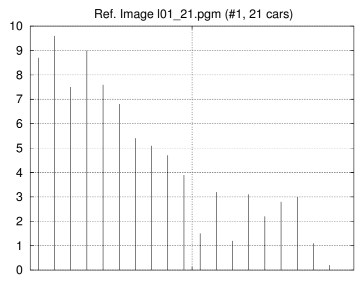
(b)



(c)



(d)



(e)

Figure 17: (a) Mean characteristics of the images; (b)–(e) absolute values of differences of mean characteristics: (b) with reference image 16b, containing 6 cars; (c) with reference image 16f, containing 9 cars; (d) with reference image 16p, containing 19 cars; (e) with reference image 16t, containing 21 cars.