[Weis93]
≡ 16979

CAR-TR-667
CS-TR-3063
May 1993

F49620-92-J-0332
DACA76-92-C-0009
IRI-9017393

GEOMETRIC INVARIANTS FOR IMAGE DATABASES

Isaac Weiss*
Walid G. Aref*,†
Ehud Rivlin*
Hanan Samet*,†,‡

*†Center for Automation Research
†Computer Science Department and
‡Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

Abstract

Image databases are conceptually much harder to deal with than conventional databases because the information that they contain consists of images, rather than alphanumeric entities. Images can be fuzzy and distorted, and they depend on the point of view from which the object is seen. Characteristics of the images which are invariant to changes in the viewpoint are presented. These characteristics can be stored as "signatures" for the objects in an atlas database, thereby permitting efficient retrieval and matching (partial or total) regardless of the viewpoint. Invariant signatures are useful because image matching is very slow in high-population image databases. They can also be indexed easily using current database technology (e.g., the B-tree). Using a sample database consisting of images of fruits, examples are given of queries in such an environment, and the advantages of using invariant signatures are illustrated.

Keywords and phrases: image databases, query processing, object recognition, invariants

The support of Isaac Weiss by the Air Force Office of Scientific Research under Grant F49620-92-J-0332; of Isaac Weiss and Ehud Rivlin by the Advanced Research Projects Agency (ARPA Order No. 8459) and the U.S. Army Topographic Engineering Center under Contract DACA76-92-C-0009; and of Walid G. Aref and Hanan Samet by the National Science Foundation under Grant IRI-9017393 is gratefully acknowledged, as is the help of Sandy German in preparing this paper.

1 Introduction

Incorporating images in database applications is a challenging problem [19]. Given an image of an unknown object, we can present the image to the database as a query: find a record in the database containing an image of a similar object, and give us other related information in the record such as the name and other attributes of the object. In this way, the system "recognizes" an object, i.e., it can provide useful information about the object just by looking at its image.

However, the construction of an image database poses special difficulties that are not encountered in most other applications. Most databases use an alphanumeric information representation of the data. This representation is ideally unique, precise and stable. Images are far from having these qualities which makes matching difficult. Some of the difficulties are: (a) images are corrupted by noise, occlusion, camera distortion, poor lighting, etc. (b) Even ideal images do not provide an adequate basis for object identification, because we want to be able to associate slightly different objects with the same identifier; for instance, we want to identify apples as apples even though all apples are not exactly alike. (c) Points (a) and (b) apply in other domains such as speech recognition [8], but there is also a problem unique to images, resulting from the fact that an object can be seen from different points of view. Each view yields a different image; we would like to store only one of them in the database, but the problem then arises of how to match the stored image with an image that was taken from a different viewpoint, if that image is presented to the database as a query. We do not deal here with the semantics of the image (e.g., [13]); we are concerned only with the shapes of the objects in the image.

In this paper we propose the use of *geometric invariants* as an alternative representation for images in image databases to speed up search and matching queries. Invariants are quantities computed from the image that remain unchanged under changes in geometric conditions such as the point of view. These quantities can be stored in the database instead of the images themselves. We call such stored data an invariant *signature* of the object. When a query image is processed, its invariant signature is matched against the stored signatures, rather than matching the images themselves. In this way we can obtain a match even if the stored data and the query represent images taken from different points of view. Similarly, we can obtain invariants to small deformations, e.g., the ones that represent the differences between different apples. One of the advantages of signatures is that they can be indexed easily using current database technology (e.g., B-trees). We compute only one signature per object, chosen to capture the features of interest in the object. We can then use any one-dimensional indexing technique to speed up access.

We can illustrate this concept with a simple example. In a simple world consisting of sticks, we want to identify a particular stick by its image. The sticks can lie in any position and orientation, but we want to recognize a stick regardless of these geometric variations. To do so we can calculate the *length* of the stick. Length is invariant to rotation and translation and therefore we can use it as an identifying signature that is independent of the position and orientation of the stick. Of course more complicated objects will require more complicated invariant quantities.

The stick example illustrates an invariant to Euclidean transformations, namely rotation and translation. Other transformations of interest are small deformations. Images also vary under larger transformations—in particular, projective transformations, which are involved in a general change of viewpoint.

The role of projections in image formation was probably first recognized by the Renaissance artist Brunelleschi in 1413 [7]. Projective invariants were a very active mathematical subject in the 19th century [15]. In the computer vision community, one invariant (the cross ratio of four collinear points) was discussed by Duda and Hart [8]. General invariants of curves and surfaces were introduced by Weiss [26], who also pointed out their utility in building image databases. This subject is now developing rapidly.

In this paper our focus is on the various issues involved in the application of geometric invariants in an image database environment. In Section 2 we present a sample database application which is amenable to the use of geometric invariants. Section 3 describes our database and the types of queries that we wish to be able to answer. Sections 4–6 are devoted to invariants. In particular, Section 4 discusses the role of invariants in image databases, Section 5 addresses the issue of which invariants to use, and Section 6 gives a non-mathematical explanation of the highlights of invariants. Section 7 contains the results of some experiments on a set of images in our example database and illustrates how invariants facilitate responding to the queries posed in Section 3. Concluding remarks are made in Section 8, and directions for future research are also briefly discussed.

2 Application

Our application domain is a database of images of fruits. Our goal is to distinguish between different types of fruits as well as to find defects in the fruits. The fruits are not constrained to lie in a particular position or to have a given orientation; and one fruit may occlude another.

Such a database has many potential uses. For example, suppose that we have a conveyor belt on which the fruits pass under a television camera. Our task may be real-time inspection, e.g. detecting the presence of spoiled or undersized fruits and also rejecting them. Alternatively, we may want to know how many pears, bananas, apples, etc. are present, or their average sizes or color characteristics.

A specific application might be to a warehouse or shipping dock. Suppose there are many different boxes of fruit and we wish to know which fruits are in which boxes, in order to register the boxes in a database. Since the names and characteristics of the fruits differ from country to country, one solution is for the database to have a field containing an image of the contents of each box. This is a self-describing database that is independent of the name; in other words, it is content-addressable. Now, we can respond to queries such as "find all boxes containing plantains that weigh over 50 pounds, that were harvested before September 1, 1992, and that have been in the warehouse for no more than one month". (Plantains are variants of bananas.) Our techniques can identify the bananas from their images.

A plausible alternative to the use of an image as a description of the contents of a box is to use a preprocessing technique to identify its contents and attach a bar code to the box. The identification can be manual or automatic. Our techniques are applicable to the latter. Nevertheless, the bar code in itself may not be enough to characterize the shipment as we don't know all of the possible queries in advance. For example, we may wish to know if some of the fruits have imperfect colors or if they have other characteristics which are not known a priori. Thus a real-time capability to understand the contents of the box is useful. Another argument against our methods is that all we have is a two-dimensional image of some of the contents of the box. However, quality control inspections do work on the same principle of just looking at a subsample. If the subsample is defective, then the entire sample is rejected. We use the same principle.

Our goal is to distinguish between the various fruits, as well as between instances of a particular fruit, by use of shape-invariant classification techniques [22]. These techniques enable us to obtain a signature in the form of a curve for each of the fruits. Occlusion is dealt with by using partial signatures. The issue of size can be dealt with by specifying the position of the camera relative to the box or conveyor belt. Factors which depend on the surface quality of the fruit (e.g., spots and blemishes) can be detected by the application of standard image processing techniques such as histograms [23].

Our usage of invariants is superior to that of others in the sense that we use more of the information available in the image. For example, other researchers working in the theory of invariants [3, 9] suggest the use of invariants as a way of matching against stored models or indexing into a database of imaged objects. However, the invariants are formed from a very limited subset of the image, thereby not permitting variations or irregularities such as those arising from noise or differences in the position of the viewer. For example, in the case of aircraft identification, Barrett [3] suggests the use of five coplanar points. We can find five points on or near a plane and then compute two invariants for these five points. The method of local invariants is much richer. Local invariants can be defined at each point of a shape, and can be used to obtain a "signature" of that shape. Thus we take into account the shape properties at every point. Being local, the invariants are much less sensitive to occlusion than global invariants.

Our techniques are also useful for queries that are based on the use of an image as part of the query. For example, suppose that we want to know if a particular fruit is present in one of the boxes in the warehouse. The query can be posed by using the picture of the fruit. Thus we are using the well-known technique of query-by-example [30] where the picture is one of the fields of the tuples stored in the database. This is especially useful if we don't know the name of the fruit. It enables us to pose the query in a language-independent manner.

As another example, suppose that we wish to find all shipments containing bananas typically found in Martinique. The easiest way to handle this query is to have an attribute in the database indicating the origin of the box. However, perhaps such bananas are now also grown in Costa Rica. By using a picture of the banana, we can find similar bananas. Thus we have a more flexible database; of course, at the added cost of processing since now we must perform the classification in real time.

Another way is to use a graphical user interface to outline the fruit. A signature based on the use of invariants can be computed from the outline and matched against the database. The use of an outline, but not an invariant-based signature is used in the QBIC system [2].

Our approach is superior to that of Jagadish [18] which only works for objects that can be decomposed into rectangles. His approach is classical in the sense that he parametrizes the image (by decomposing it into a small number of rectangles) and then maps the object's parametrized representation into a point in a higher-dimensional space. This technique does not take full advantage of the shape domain. In particular, the shape vocabulary is limited to orthogonal rectangles. It also does not lend itself to queries about the relationships of the objects to their neighbors. For example, it is difficult to find the nearest banana to a given point using such an approach (but see [16] for a solution to such problems in the narrower domain of line segments). In [11] some indexing techniques are proposed but the viewpoint problem is ignored.

3 Database Description and Sample Queries

In this section we describe a possible database as well as some sample queries. Associated with each shipment in the database is a shipment identifier and a time-stamp which indicates the shipment's arrival time at the warehouse. Our database consists of two main parts: static and dynamic. The static part is an atlas of fruits where information is stored about all the types of fruits known to the database. The dynamic part contains the information that is relevant to the day-to-day transactions involving the import, export, and handling of fruits as described above. The types of queries that are expected are described below using an appropriate adaptation of SQL (e.g., [24]).

The information in the atlas facilitates the recognition and classification processes. Each entry in the atlas corresponds to one type of fruit. For each fruit type in the atlas, the following attributes are stored: fruit-name, fruit-class, country of origin, color-code, quality-code, fruit-size, invariant-signature, and image. One possible schema is:

```
create table fruit-atlas (
fruit-name VCHAR(30),
fruit-class VCHAR(30),
country-of-origin VCHAR(30),
color-code INTEGER,
quality-code INTEGER,
fruit-size FLOAT,
invariant-signature SIGNATURE,
image IMAGE);
```

`invariant-signature` is of type `SIGNATURE` while `image` is of type `IMAGE`. Both of these types are abstract data types. The `IMAGE` data type can be implemented as a variable-length two-dimensional array, or as a long field [20, 21].

An example entry in the fruit-atlas database is
`(plantain, banana, Brazil, 112(level of yellow), 9, 12, 12-12-10-10, image-1)`.
This entry corresponds to `plantain bananas` imported from `Brazil` with color code `yellow`, average quality `9/10`, average size `12` inches, and invariant signature
`12-12-10-10`.

The interpretation and properties of the signature are discussed further in Section 4. The atlas is populated by a learning system which computes the relevant signatures and associates them with the corresponding instances of each fruit type. The signature is the classifier for the on-line fruit recognition system. The signature attribute is assumed to be unique for each entry in the database and hence can be used as a primary key. `image-1` is the value of the attribute image. An example of this for a pear is given in Figure 3.

Notice that the signature is independent of the shape or boundary of the fruit. It reflects neither the actual size of the fruit nor its color. The values of these attributes are stored explicitly with the fruit entry (tuple). In order to deal with issues of quality control, we must have a specification of acceptable, as well as unacceptable, defects for each fruit. This information is stored in the relation `defects`. One possible schema is:

```
create table defects (
defect-code VCHAR(30),
defect-name VCHAR(30),
fruit-signature SIGNATURE,
defect-signature SIGNATURE,
image IMAGE);
```

The dynamic portion of the database is composed of the following relation, termed `shipment`:

```
create table shipment (
shipment-id INTEGER,
time-stamp TIME,
image IMAGE),
invariant-signature SIGNATURE,
fruit-color COLOR,
fruit-size FLOAT,
defect-code SET OF INTEGER);
```

The shipment relation is populated by a realtime transaction that takes a picture of each box, and assigns it a `shipment-id` and a `time-stamp`. The picture is stored in the attribute `image`, and is processed further to generate the signature, size, color, and the defects of the fruit associated with it. In order to identify the fruit type in a given shipment, the signature is matched with the atlas (through an index on the signature attribute of the atlas).

Let us now consider a few example queries.

Query 1: Select all possible types of bananas and their images.

```
select into bananas all
from atlas
where class = "banana"
```

Query 2: What fruits arrived in today's shipments?

```
select distinct a.name
from atlas a, shipments s
where s.time-stamp between TIME("0000/5/3/92") and TIME("2400/5/3/92")
  and s.signature = a.signature
```

Query 3: Find all the defective fruits (i.e., with black spots) in today's morning shipment.

```
select into defective  s.box-id s.time-stamp a.country-of-origin a.fruit-name
from atlas a, shipments s, defects d
where s.time-stamp between TIME("1030/5/3/92") and TIME("1430/5/3/92")
  and s.defect-code = d.defect-code
  and ( d.defect-name = "spots"
        or s.color = "black" )
  and s.signature = a.signature
```

Query 4: Find out how many boxes of plantain bananas arrived today.

```
select count(box-id)
from atlas a, shipments s
where s.time-stamp between 0000 and 2400
  and a.fruit-class = "banana"
  and a.fruit-name = "plantain"
  and a.signature = s.signature
```

Query 5: Find out which fruit is in the given image.

```
select all
from atlas a
where a.signature = SIGNATURE(InputImage)
```

Query 5 is evaluated by computing the signature of the given image InputImage using the scalar function SIGNATURE, and then using the index on a.signature to retrieve the matching fruit from the atlas. Notice that there is no need to access or process a.image in this query since this attribute (image) is preprocessed and only its signature is significant. Also notice that InputImage and a.image don't have to match exactly because the signature filters out scaling, rotation, and translation.

Query 6: Find how many boxes were delivered today with the same defect as the one shown in the given image.

```
select count(box-id)
from atlas a, shipments s
where s.time-stamp between TIME("0000/5/3/92") and TIME("2400/5/3/92")
  and s.defects in DEFECTS(InputImage)
```

DEFECTS is a function whose value is a table [14]. One way to evaluate query 6 is to use the signature of InputImage and to access the shipment relation s through s.signature to select the boxes with the same signature, and then to match the defects found in InputImage with the defect attribute of each qualifying shipment. Only the shipments with the time stamp between 0000/5/3/92 and 2400/5/3/92 are reported.

4 The Role of Invariants in Image Databases

It can be argued that searching a database for matching images is equivalent to searching for invariants. This is also a way of characterizing the task of object recognition in computer vision and our discussion makes heavy use of this analogy. Given an image of an object, we want to extract one unique invariant. Given another image of the same object, differing from the first by, e.g., viewpoint, we want to extract the same unique descriptor. To do this, we have to eliminate in some way the effect of the transformations that gave rise to the differences between the images.

There are several methods of eliminating transformations between images. The simplest is by performing every possible transformation of one image and seeing if any of its transformed versions matches the other image. For instance, in template matching [1], it is assumed that a template and an image differ only by translation, and the template is moved pixel by pixel over the image until a match is found. However, when more complicated transformations are involved, such as rotation, projection, etc. this search space becomes overwhelmingly large.

To reduce the search space, "invariant features" can be used. These are features in the image that stay invariant under some transformation and can be matched directly between the two images. For example, an edge remains an edge, so edges can be used for matching. The problem here is that the kinds of features usually used do not have much distinctiveness. Any edge in one image can match any edge in the other. This leads to a *correspondence problem*, which can easily lead to a combinatorial explosion. Invariant constraints [10] can also be used but they still leave a large space to search in.

Other methods aimed at viewpoint invariance have their own drawbacks. Fourier descriptors are not fully invariant and suffer from occlusion problems. Hashing methods such as the Hough transform [1, 17] break down when a large number of parameters is involved.

The correspondence problem can be solved by using more distinctive invariant descriptors—that is, descriptors that are invariant only to the transformation we are interested in and not to others. For instance, a shape descriptor of a fish should be distinct from a descriptor of a frog—that is, it should not be invariant to a transformation that maps the shape of the fish into that of the frog. Edges, of course, are invariant to this since they can appear in both shapes. However, they are "too" invariant—that is, they are invariant to too wide a set of transformations. Thus, we must try to find features that are invariant only to the transformations that we want to eliminate and to no others, so they are distinctive enough to match without ambiguity.

Change in the point of view is only one kind of geometric transformation that images can undergo. For instance, we would like to identify an object as a "fish" even if the particular example of a fish we are looking at is somewhat thinner or fatter than some standard fish. In this case we need invariants to *deformations*, i.e., quantities that will not change under a not-too-great deformation of the object. It is again important not to seek invariance to transformations that are too general, because then the descriptors will blur the distinction between different objects.

A fundamental question immediately arises: what transformations do we want to eliminate? When do we decide that two images come from the same object, even though they are different? Viewpoint change is one example; other transformations will probably depend on the types of objects in question.

Another consideration in choosing the kind of invariance we need is that the larger the set of transformations, the harder it is to extract meaningful distinctive quantities that are invariant to it. (For example: distance is a Euclidean invariant but is not preserved under projection.) Yet the need for invariants is much more acute, because the larger set of transformations has more unknown parameters and requires a search in a much bigger space. This consideration thus leads to the same conclusion as the distinctiveness argument: we have to find optimal invariants, i.e., ones that will stay invariant under the set of transformations that we want to eliminate, but not under a larger set.

A paradigm for matching in image databases can thus include the following:

1. Identify the transformations that an image can undergo and still describe the same object—that is, the transformations that we want to eliminate for particular classes of objects.
2. Find descriptors that are invariant to these transformations but not to others.
3. Use these descriptors to index shapes and match them.

In the next section we discuss item 1 above, while items 2 and 3 are dealt with in the remaining sections.

5 Which Invariants?

In this paper, we only deal with purely geometric invariants—that is, ones that can be calculated from the shape alone. Other surface properties such as shading, reflectance, color, etc. can also be considered as invariants, subject to the same considerations as above, but are beyond the scope of this paper.

The most obvious useful invariants are the ones that are invariant to the Euclidean transformations (i.e., translation and rotation). A simple example is the length of a rod, which is invariant under rotation. In a simple world consisting of rods that lie in a plane, and images that can only rotate, we can identify a particular rod by measuring its length on the image and comparing it to a database of rod lengths. The rod's orientation is irrelevant and can be ignored. As another example, when a 2-D curve is rotated or translated in the plane, its curvature at each point does not change. Thus curvature is an invariant of the Euclidean transformations. It is common to plot the curvature of such a curve as a function of its arc-length (which is invariant up to a starting point [12]) to obtain a 2-D Euclidean invariant representation of the curve. Curvatures of surfaces have also been used when they can be measured (e.g., from range data [4]).

The formation of images in general involves a larger set of transformations (including the Euclidean ones). A projective transformation, for example, is more general than a Euclidean transformation, and includes projections between planes that are not parallel to each other. The number of free parameters in this case is eight[1], so finding the correct point of view can involve a search in an 8-dimensional space! Clearly projective invariants, namely quantities that are unchanged under this transformation, are of crucial importance.

When enlarging the transformation set, the problem arises that the invariants of the smaller set do not remain invariant. The length of a rod is no longer invariant under projective transformation. Similarly, an oblique view of a circular disc yields an ellipse, and obviously neither arc-length nor curvature is preserved under such projections [25].

To find invariants of larger sets, we have to extract more information from the image. While finding length requires two points, a similar projective invariant needs four, so we need to extract more data from the image to obtain reliable results. This is more than offset computationally by the enormous savings resulting from the elimination of the search. However, it does lead us to conclude that we should not enlarge the transformation group beyond what is absolutely necessary.

Projective transformations (termed *projectivities*) are the smallest group that includes all possible viewpoint-related changes in images, and therefore we concentrate on them. Apart from the invariants issue, using projective geometry can unify and simplify the treatment of perspective and orthographic projections, which are often treated separately [1].

The most readily useful projectivities are the ones operating on a 2-D plane. One view is sufficient to reconstruct a planar shape (except for the projectivity) [29]. Therefore invariants by themselves are sufficient as a means for indexing and recognizing planar shapes. They are also applicable to 3-D objects, since many objects contain planar shapes, such as facets, symmetry planes, etc., which are generally projected onto the image as planes. In addition, small areas of a 3-D surface can be approximated as planar. Thus, 2-D projective transformations and their invariants can be used for recognition of many 3-D objects.

Smaller subsets of the projective transformations are often quite useful. When the object is far from the camera, we can assume that the projection rays are nearly parallel, which defines the affine transformations. If we can find one feature point that can be regarded as unchanged in the projection, we have a perspective transformation. Euclidean transformations are a common subset of both the affine and perspective transformations.

In 3-D, we rarely need to consider a full projection. A surface in 3-D can be translated, rotated or perhaps scaled, but not projected. However, 3-D projective invariants of curves and surfaces do exist and they are summarized by Weiss [3, 6, 26]. The Euclidean and affine 3-D invariants have the same role of indexing of 3-D shapes as the projective invariants have in 2-D.

[1]Translation, rotation, scaling (horizontal and vertical), shear, and slant (horizontal and vertical).

The case of projecting a 3-D object into a 2-D image is of a different nature. In this case, true invariants cannot be found because the depth information is missing and cannot be recovered by purely geometrical methods. Additional, "model-based" knowledge is needed to reconstruct the missing information, and this is beyond the capacity of invariants alone. However, invariants can be very useful in combination with additional information.

The image of an object can change on account of more than just a change in the viewpoint. Thus there is a need for more general geometric invariants. As mentioned before, it is interesting to deal with invariants of small deformations, e.g., ones that change the shape of a fish somewhat but still leave the general shape intact. Here we are only interested in small deformations because if the distortion is too large it may change the object beyond recognition. Invariants for general deformations probably do not exist because if they were invariant to any kind of change, then they would be meaningless. However, we can find "quasi-invariants", namely quantities that change slowly relative to the transformation itself. Thus, if the deformation is small, then the change in the quasi-invariant will be even smaller and it will be observed as an invariant for all practical purposes. If the distortion is very large, then we may have a different object altogether and the quasi-invariant is different too.

6 Highlights of Invariants

Deriving invariants can be quite mathematically involved and we cannot discuss them fully in this short paper. We will only give here a number of examples of affine and projective invariants as well as quasi-invariants. For more details see the review by Weiss [27].

There are two main kinds of invariants: global and local. The calculation of global invariants requires knowledge of the entire shape. For example, we need a whole contour to find area, a Euclidean invariant. Moments and Fourier coefficients are other examples of global shape descriptors which have some invariant properties. Global invariants are relatively easy to calculate but they are sensitive to occlusion. That is, if part of the shape is missing from the image or is occluded by another object, we obtain a totally incorrect value for the descriptor.

Local invariants avoid the occlusion problem by performing the calculation pointwise, i.e., in a small neighborhood around each point of the visible contour of the shape. For example, it is quite common to plot the curvature of a contour with respect to the arclength—that is, the length along the contour from some starting point to some given point. This allows to deal with "incomplete" queries, in which part of the information expected in the query is missing (due to occlusion) but there is still enough information to retrieve the desired record. Denoting the curvature and arclength by κ and s respectively, we obtain a curve of $\kappa(s)$ representing the visible contour. Both arclength and curvature are invariant to Euclidean transformation, and thus we obtain an "invariant signature" that can identify the curve. This curve can be stored in a database and it will match the signature of a similar object (contour) presented as a query, even if the query object is moved or rotated with respect to the object originally stored in the database.

It is desirable that images be invariant to transformations more general than the Euclidean ones. Below we give an example of affine and projective local signatures of objects. An affine transformation is simply a general linear transformation in any coordinate space. Since images are two-dimensional entities, most of the work on using invariants has been limited to the plane. An affine transformation in the plane has six degrees of freedom: the three Euclidean ones (translation in x and y directions and rotation), scaling (independently in the x and y direction) and shear. A projective transformation is more general. It includes the foreshortening of images resulting from slanting one plane with respect to the other. The slanting can occur in either the x or y directions (or a combination of them), and thus we have two additional degrees of freedom.

We have already mentioned that length is a Euclidean invariant. Others are area, angle, and curvature (the latter being a local invariant). None of these stay invariant under the more general affine transformation. However, it is easy to show that *ratios* of areas are affine invariants. Ratios of lengths are invariant only if the line segments are collinear. There are also local affine invariants, the so called affine curvature and the affine arclength [12]. Local invariants are obtained from a curve (or surface) point and its infinitesimal neighborhood, rather than from discrete features. These can be used to obtain an affine invariant signature of a curve, by plotting the affine curvature against the affine arclength. The signature can then be used to identify the curve as described earlier. The curve can be the boundary of the object or some part of it.

The easiest way to show the invariance of the ratios is by using the method of determinants. Under a linear transformation, any determinant is multiplied by the Jacobian of the transformation (i.e., the determinant of the transformation itself), and that makes any determinant a "relative" invariant. The ratio of determinants eliminates the multiplicative factor and is thus an invariant. Areas and lengths can be represented as determinants and thus their ratios are invariant.

In the projective case, the ratios are no longer invariant. However, the so called *cross ratios* (or ratios of ratios) are. Figure 1 shows four collinear points projected from one line to another. The cross ratio of the lengths of the line segments is

$$\frac{l_{12}l_{34}}{l_{13}l_{24}} = \frac{(x_1 - x_2)(x_3 - x_4)}{(x_1 - x_3)(x_2 - x_4)},$$

where the x_i are the coordinates of the point along the line and the l_{ij} are lengths. This cross ratio is invariant to projection. Similarly, given five points in the plane (no three of which are collinear), the cross ratio of the areas of the triangles defined from triples of the points is invariant. These relations can be proven using determinants defined in so-called homogeneous projective coordinates.

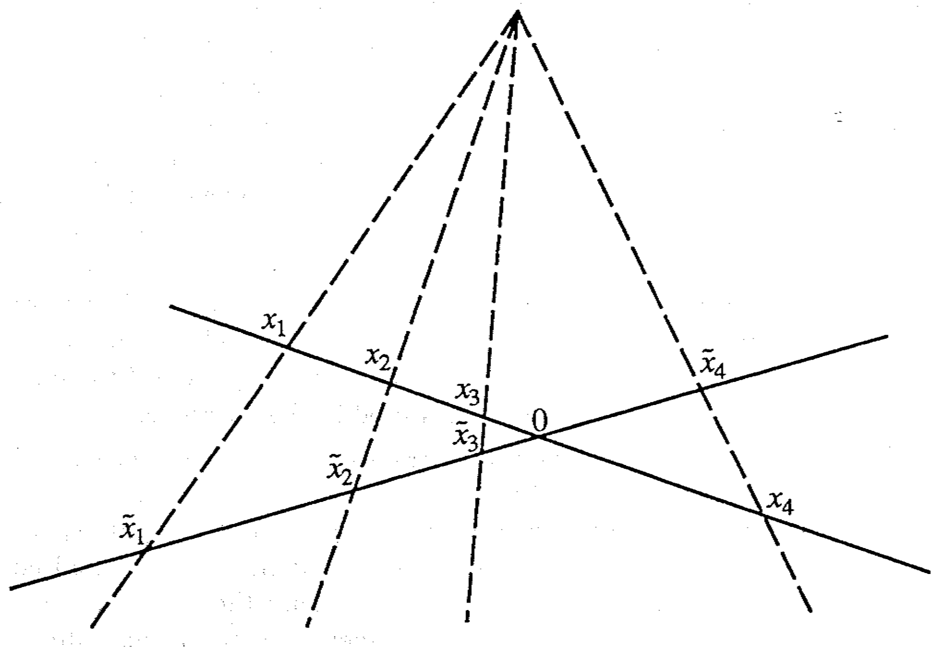


Figure 1: Projection of four points from one line to another.

As for local invariants, the situation is more complicated than in the affine and Euclidean cases, and the method of determinants cannot be used. The difficulty lies in the non-linearity of the transformation. However, local projective invariants have been found by several methods [28, 29]. The easiest way is probably the canonical method. The basic idea is to transform the given coordinate system to a "canonical", or standard system, which is determined by the shape itself. Since this canonical system is independent of the original system, it is invariant. All quantities defined in it are thus invariant.

This is easy to illustrate in the Euclidean case. To find an invariant at a given point on a curve, we change the x, y axes so that the new x-axis is tangent to the curve at that point. We thus have $\bar{y}' = 0$, while the second derivative $\bar{y}''$ at this point is now the curvature. It is invariant since we obtain this canonical system regardless of the coordinate system with which we start. We see that by determining some of the properties of the system, the others are also determined and become invariant. We have generalized this process to the affine and projective cases and found two local invariants at each point [22]. These can be plotted against one another to obtain an invariant signature curve.

The next problem is finding invariants of small deformations, as mentioned before, so we can deal with slightly different objects such as two apples. Here we can use the concept of *quasi-invariants*, introduced by [5] Binford (albeit in a different context). Unlike strict invariants, which are absolutely unchanged under a transformation, the quasi-invariants do change but at a much slower rate than the transformation itself. Mathematically, the first derivative of the quasi-invariant (with respect to the transformation) vanishes and the change is only in the second order. If we keep the higher terms of the Taylor expansion of the transformation small, than the high order terms in the invariants are also small and their change is slow.

It is easy to prove that affine invariants are quasi-invariants with respect to small deformations. Any deformation can be expanded in a Taylor series, and the first order terms of the expansion are of course linear. The affine invariants are already invariant to this linear part, and only the small higher order parts can affect it. Thus the affine invariants meet the test of quasi-invariants of the more general (small) deformations, namely they change only slightly. Thus, we use the affine invariants as signature for classes of objects such as apples. In the following we use the boundaries of objects as our curves. The quasi-invariance not only assures that we get the same signature for slightly different apples but also accounts for the fact that different points of view make the visible boundary somewhat different.

7 Experiments

Our sample database consists of a collection of fruits. The database is small because the implementation is still in the developmental stage, but it seems to be large enough to provide a proof of the basic principles. In the following examples we make use of the occluding boundaries of the fruits, i.e., the boundaries visible to the eye as we look from a particular point of view. Unlike a rigid object which stays the same regardless of the point of view (even if it looks different), the occluding boundary changes with the change in the point of view. That is, not only the projection of the boundary in the image changes, but the boundary itself changes; we see a different boundary when we change our viewpoint. Nevertheless, the basic shapes of the fruits remain more or less similar. We can recognize a pear shape regardless of which boundary we look at, except for some degenerate points of view.

Because the boundary changes, we cannot expect to find strict invariants to changes in point of view. Such strict invariance can only hold for rigid two-dimensional objects. However, we can expect to find "quasi-invariants", i.e., quantities that change more slowly than the changing point of view. The signatures of fruits seen from different angles will not be identical but will be similar regardless of the point of view. We used the affine and projective invariants discussed above as our quasi-invariants. We compared the signatures of pears, bananas and apples. The following figures illustrate the resulting signatures.

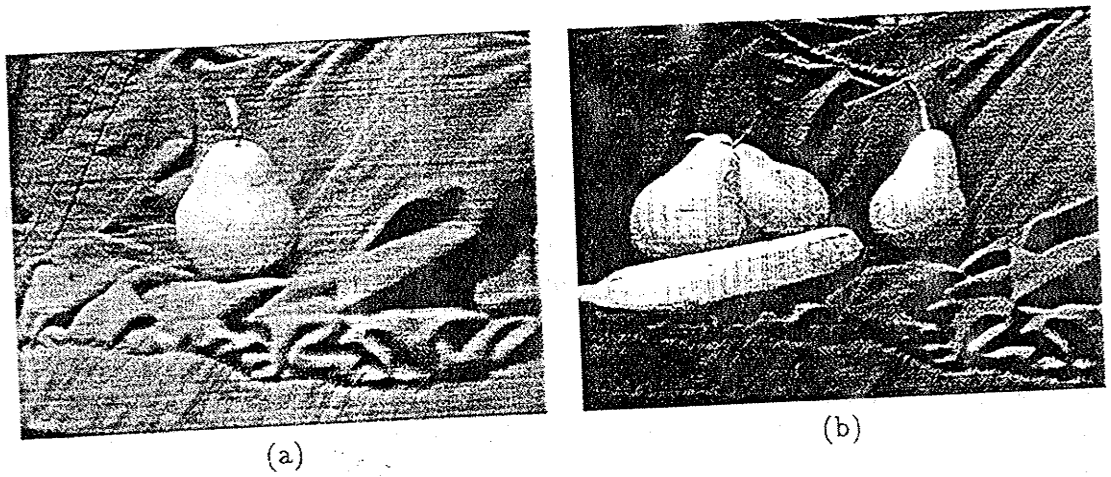


Figure 2: Two views of pears.

Figure 2 shows two views of pears. The projective signature of the pear in Figure 2a was compared with the one in Figure 2b. Some distortion effects can be seen. Figures 3 and 4 show the projective signatures for the pears in Figure 2a and 2b, respectively. Notice the close similarity between the two signatures.

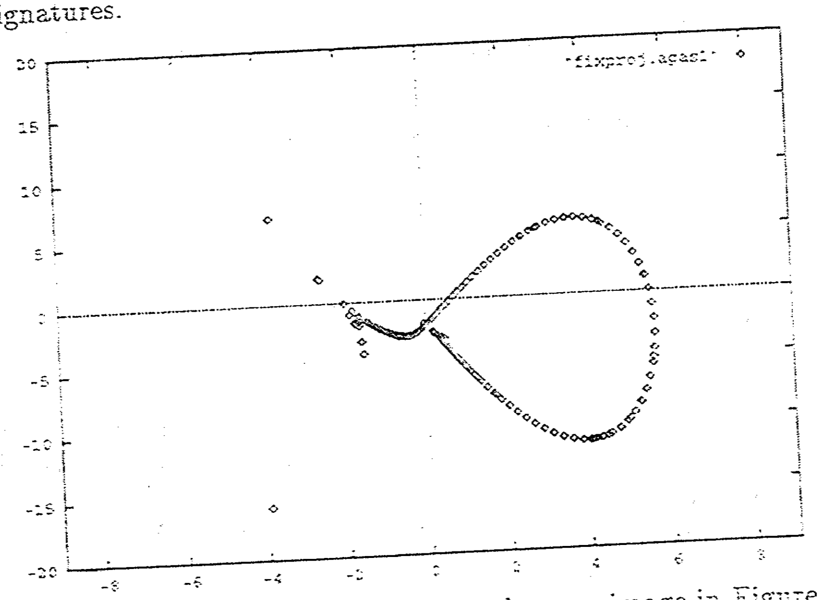


Figure 3: The projective signature for the pear image in Figure 2a.

Figure 5 shows the affine signatures for the pears in Figure 2. The affine signature is invariant to affine transformations like scale, rotation, translation, shear, and skew. Unlike the more general projective signature, it does not remain invariant under perspective projection. However, the affine signature is simpler to calculate than the projective one. Figure 6 shows a good match between

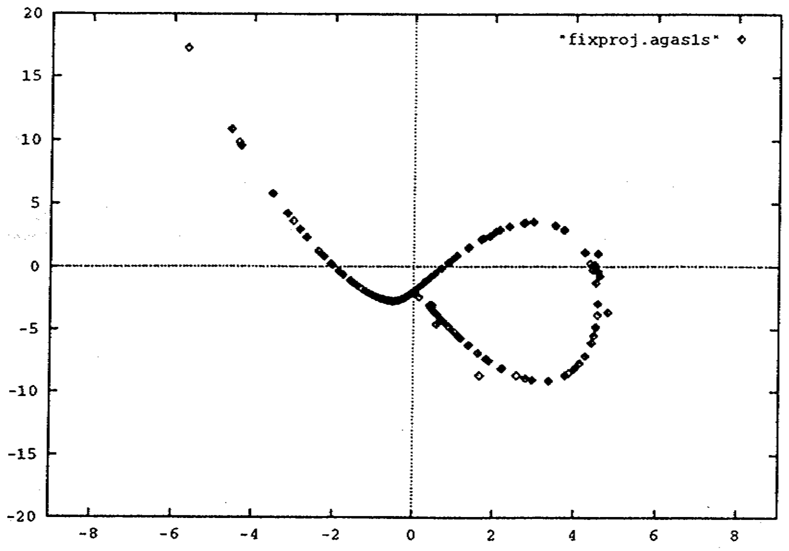
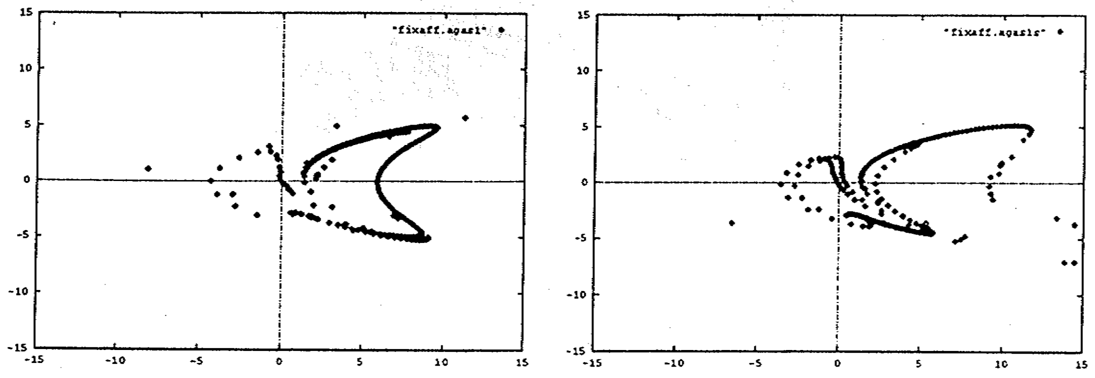


Figure 4: The projective signature for the pear in Figure 2b.

the signatures for the two pears in Figure 2. The difference between the affine signature of the pear in Figure 2a and that of the banana in Figure 7 is shown in Figure 6b. The signatures are superimposed on each other.

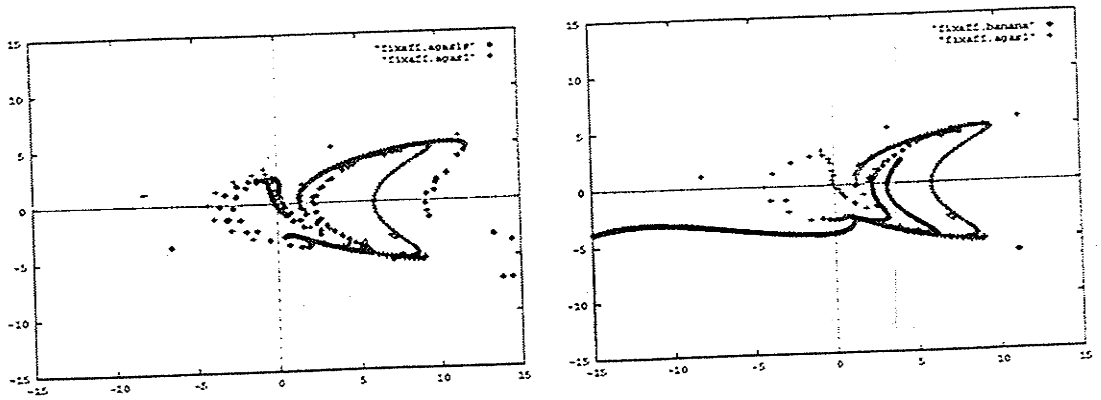


(a) (b)

Figure 5: Affine signatures for the pears in Figure 2.

Figure 7 shows a different fruit, a banana. We computed the projective signature for the banana and compared it with the projective signature of the pear in Figure 2a. The projective signature for the banana is presented in Figure 8a, and the result of the comparison (via superposition) is shown in Figure 8b. It should be emphasized that a bigger banana would result in an identical projective signature. This is because the projective signature is invariant to projective transformations, which include scaling.

The projective signature of the apple in Figure 9a was compared with the projective signature obtained from the apple in the middle of the box of apples in Figure 9b. The signatures for the two apples are shown in Figure 10. Figure 11a is the result of superimposing the signatures of the two apples in Figure 9, while Figure 11b is the result of superimposing the signatures of the apple in Figure 9a and the banana of Figure 7. Figure 12a shows a box of bananas, while Figure 12b is



(a) (b)

Figure 6: (a) The local affine signatures for the pears in Figure 2 are presented on top of each other. (b) The signature of the pear in Figure 2a is compared with the affine signature of a banana.

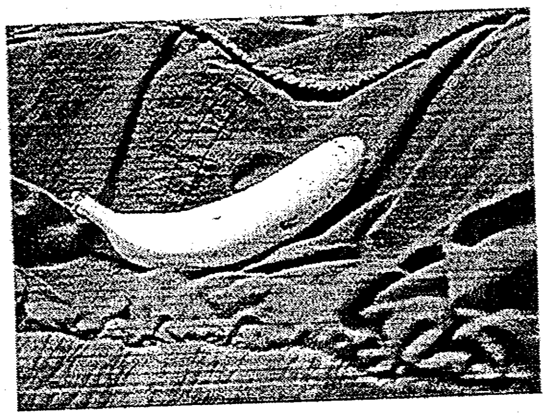
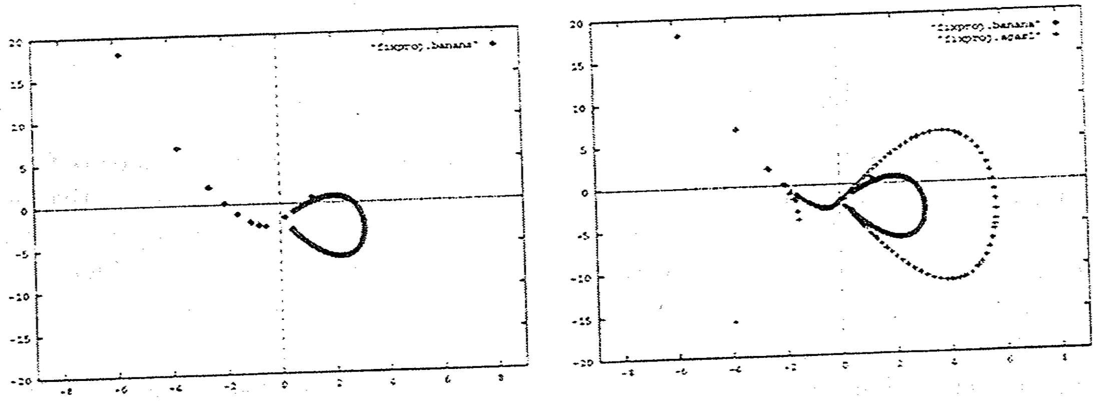


Figure 7: A banana.

the projective signature of the banana in the middle of the box.



(a) (b)

Figure 8: (a) The projective signature for the banana. (b) The projective signature for the banana presented on top of that for the pear in Figure 2a.

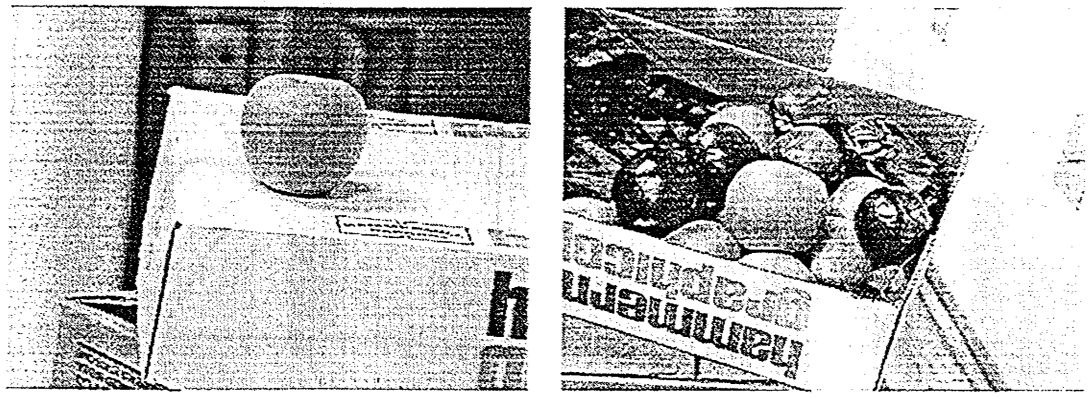

(a) (b)

Figure 9: Sample apples.

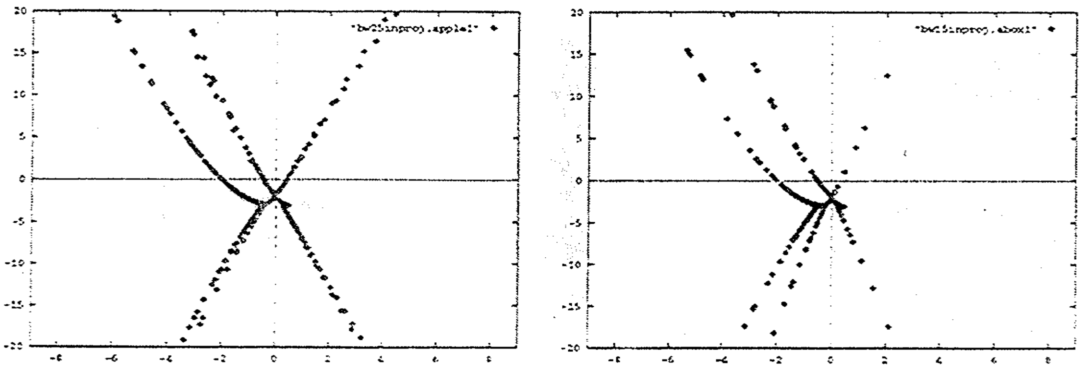

(a) (b)

Figure 10: Projective signatures for the apples in Figure 9.

The conclusion that can be drawn from the figures is that the signatures of similar fruits are similar, while fruits of different species have different signatures. Thus we can distinguish, for example, between a pear and a banana or a banana and an apple, but not between different pears. The ability to do this enables us to answer questions like query 2 ("What fruits arrived in today's shipments?"). All we need in the signature part is the ability to distinguish between the signatures of different kinds of fruit. The same reasoning also enables us to respond to query 5: "Find out which fruits are in the given image".

The use of projective and affine invariant signatures gives us the ability to classify fruits into apples, pears, bananas etc., and to do it regardless of the point of view from which the fruit was viewed. In the future, we plan to extend our implementation by increasing the expressive power of our shape descriptors so that we will be able to distinguish between different species of bananas, between a good and a defective banana, etc. Thus answering queries like: "Find out how many boxes of plantain bananas have arrived today" (query 4) is part of our future research. We did not address defect analysis. It can be accomplished using different methods of image analysis (e.g., histogram analysis, etc.). Our main contribution here is the incorporation of shape analysis into an image database.

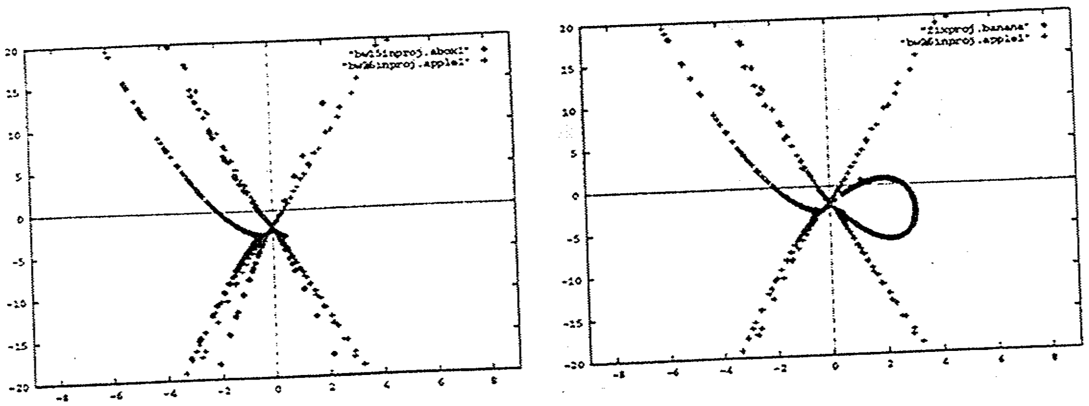

(a) (b)

Figure 11: (a) The result of superimposing the two projective signatures of the apples in Figure 9; (b) the result of superimposing the signature of the apple in Figure 9a with that of the banana in Figure 7.

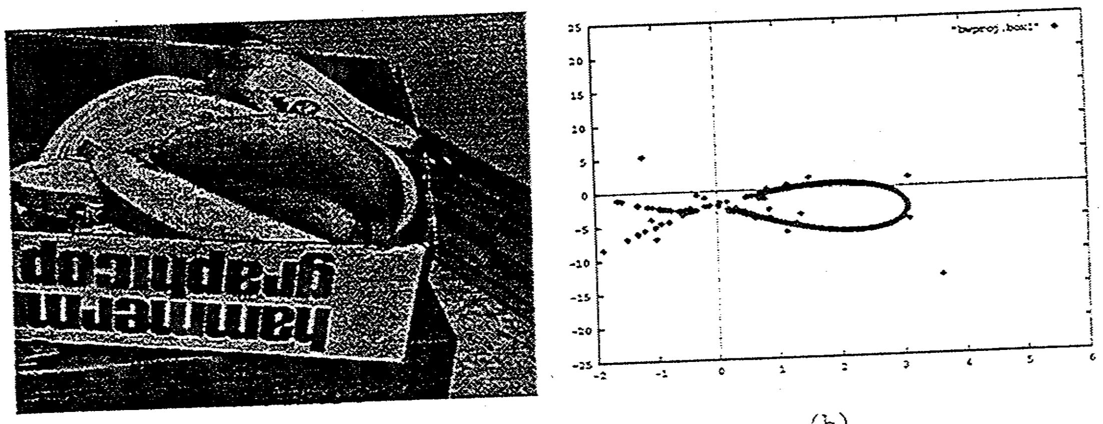

(a) (b)

Figure 12: (a) A box of bananas; (b) the projective signature of the banana in the middle of the box.

8 Concluding Remarks

Image databases have many potential applications, but their development has been very slow because of the difficulties inherent in representing images. In this paper we have addressed some of these difficulties by using the theory of geometric invariants. One of the major obstacles to the creation of image databases is that the same object yields different images when seen from different viewpoints, and this makes the matching of stored and observed images very difficult. We have overcome this problem by using invariant signatures of the images rather than the images themselves as identifiers of the objects. We dealt with invariants to geometric transformations such as rotation, projection, similarity, etc and also small deformations. More general types of invariance are also of interest—e.g., invariance to illumination or color. These invariants serve to represent the essential information in the image and index it, thereby eliminating irrelevant information, which is not part of the object itself, such as the viewpoint. Some of these methods may find use in other types of non-alphanumeric databases as well, such as voice databases.

References

[1] D. Ballard and C. M. Brown. *Computer Vision.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[2] R. Barber, W. Cody, W. Equitz, M. Flickner, E. Glasman, W. Niblack, and D. Petkovic. Query by image content (qbic) status as of 8/92. Technical Report RJ 8961, IBM Research Division, San Jose, CA, September 1992.

[3] E. Barrett, P. Payton, Haag N., and M. Brill. General methods for determining projective invariants in imagery. *CVGIP:IU*, 53:45–65, 1991.

[4] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17:75–145, 1985.

[5] T.O. Binford. Inferring surfaces from images. *Artificial Intelligence*, 17:205–244, 1981.

[6] A. Bruckstein and A. N. Netravali. Invariant signatures for space curve recognition. Technical Report, AT&T Bell Laboratories, 1992.

[7] F. Brunelleschi (1377–1446). *The Complete Works.* Rizzoti, New York, 1981.

[8] R.O. Duda and P.E. Hart. *Pattern Recognition and Scene Analysis.* Wiley, New York, 1973.

[9] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3-d object recognition and pose. *IEEE Trans. PAMI*, 13:971–991, 1991.

[10] W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. PAMI*, 9:469–482, 1987.

[11] W.I. Grosky and R. Mehrotra. Index based object recognition in pictorial data management. *CVGIP*, 52:416–436, 1990.

[12] H. Guggenheimer. *Differential Geometry.* Dover, New York, 1963.

[13] A. Gupta, T. Weymouth, and R. Jain. Semantic queries with pictures: the vymsys model. In G. Lohman, editor, *Proceedings of the Seventeenth International Conference on Very Large Databases*, pages 69–79, Barcelona, Spain, September 1991.

[14] L. M. Haas, W. Chang, G. M. Lohman, J. McPherson, P. F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. J. Carey, and E. Shekita. Starburst mid flight: As the dust clears. *IEEE Trans. Knowledge and Data Engineering*, 2:143–160, 1990.

[15] D. Hilbert. Über die vollen Invariantensysteme. *Mathematische Annalen*, 42:313–373, 1893.

[16] E. G. Hoel and H. Samet. Efficient processing of spatial queries in line segment databases. In O. Günther and H. J. Schek, editors, *Advances in Spatial Databases—2nd Symposium, SSD'91*, pages 237–256, Berlin, Germany, 1991. (Springer-Verlag. Lecture Notes in Computer Science 525.)

[17] H. V. Jagadish. On indexing line segments. In D. McLeod, R. Sacks-Davis, and H. Schek, editors, *Proceedings of the Sixteenth International Conference on Very Large Data Bases*, pages 614–625, Brisbane, Australia, August 1990.

[18] H. V. Jagadish. A retrieval technique for similar shapes. In *Proceedings of the SIGMOD Conference*, pages 208–217, Denver, CO, May 1991.

[19] R. Jain. NSF workshop on visual information management systems, 1992.

[20] T. J. Lehman and B. G. Lindsay. The Starburst long field manager. In P. M. G. Apers and G. Wiederhold, editors, *Proceedings of the Fifteenth International Conference on Very Large Databases*, pages 375–383, Amsterdam, The Netherlands, August 1989.

[21] L. Lu, B. C. Ooi, A. D'Souza, and C. C. Low. Storage management in geographic information systems. In O. Günther and H. J. Schek, editors, *Advances in Spatial Databases—2nd Symposium, SSD'91*, pages 451–470, Berlin, Germany, 1991. (Springer-Verlag. Also Lecture Notes in Computer Science 525.)

[22] E. Rivlin and I. Weiss. Local invariants for recognition. Technical Report CS-TR-2977, University of Maryland, 1992.

[23] A. Rosenfeld and A. C. Kak. *Digital Picture Processing.* Academic Press, New York, second edition, 1982.

[24] N. Roussopoulos, C. Faloutsos, and T. Sellis. An efficient pictorial database system for psql. *IEEE Trans. Software Engineering*, 14:639–650, 1988.

[25] C.E. Springer. *Geometry and Analysis of Projective Spaces.* Freeman, San Francisco, 1964.

[26] I. Weiss. Projective invariants of shapes. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1125–1134, 1988.

[27] I. Weiss. Geometric invariants in object recognition. Technical Report CS-TR-2942, University of Maryland, 1992.

[28] I. Weiss. Noise resistant invariants of curves. *IEEE Trans. PAMI*, 1993, to appear.

[29] E. J. Wilczynski. *Projective Differential Geometry of Curves and Ruled Surfaces.* Teubner, Leipzig, 1906.

[30] M. M. Zloof. Query by example: A data base language. *IBM Systems Journal*, 16:324–343, 1977.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE May 1993	3. REPORT TYPE AND DATES COVERED TECHNICAL

4. TITLE AND SUBTITLE GEOMETRIC INVARIANTS FOR IMAGE DATABASES	5. FUNDING NUMBERS F49620-92-J-0332 DACA76-92-C-0009
6. AUTHOR(S) Isaac Weiss, Walid G. Aref, Ehud Rivlin, Hanan Samet	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Automation Research University of Maryland College Park, MD 20742-3275	8. PERFORMING ORGANIZATION REPORT NUMBER CAR-TR-667 CS-TR-3063
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NM, Bolling AFB, Washington, DC 20332-6448 ARPA, 3701 North Fairfax Dr., Arlington, VA 22203-1714 U.S. Army Topographic Engineering Center Fort Belvoir, VA 22060-5546	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Image databases are conceptually much harder to deal with than conventional databases because the information that they contain consists of images, rather than alphanumeric entities. Images can be fuzzy and distorted, and they depend on the point of view from which the object is seen. Characteristics of the images which are invariant to changes in the viewpoint are presented. These characteristics can be stored as "signatures" for the objects in an atlas database, thereby permitting efficient retrieval and matching (partial or total) regardless of the viewpoint. Invariant signatures are useful because image matching is very slow in high-population image databases. They can also be indexed easily using current database technology (e.g., the B-tree). Using a sample database consisting of images of fruits, examples are given of queries in such an environment, and the advantages of using invariant signatures are illustrated.

14. SUBJECT TERMS Image databases, invariants, object recognition, query processing	15. NUMBER OF PAGES 22
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR