

# NOTES ON OBJECT REPRESENTATION AND RECOGNITION

## 1. BASICS OF DIFFERENTIAL GEOMETRY

Everyone has an intuitive idea of what curves and surfaces are. However, we often use these concepts in a more formal setting, and thus the need for mathematically precise definitions of these terms.

**Definition 1.1.** A *parametrized curve*  $\alpha$  is a mapping (function) from a real interval to the  $n$ -dimensional real-space:

$$\alpha : I \rightarrow \mathbb{R}^n, \text{ where } I \text{ is a real interval } \subseteq \mathbb{R} \text{ and } n \in \mathbb{N}$$

Similarly, we define a *parametrized surface*  $s$  as being a mapping from a subset of the 2D plane to the  $n$ -dimensional real-space:

$$s : P \rightarrow \mathbb{R}^n, \text{ where } P = I \times J \text{ for real intervals } I \text{ and } J, \text{ and } n \in \mathbb{N}$$

For example,  $\alpha(t) = (x(t), y(t))$ ,  $t \in [0, 1]$  is a parametrized curve in 2 dimensions. Remark that a curve or surface can cross itself at any number of points.

**Definition 1.2.** The *trace* of a curve  $\alpha$  is the set of points through which it passes.  $\text{trace}(\alpha) := \alpha(I)$ , the image of the map  $\alpha$ .

Note that a curve  $\alpha$  uniquely determines its trace. On the other hand, a set of points does not determine uniquely a curve. For example, if we have a curve  $\alpha$  such that its trace is  $S$ , then the curve  $-\alpha$ , obtained by reversing the orientation of  $\alpha$ , is not the same curve as  $\alpha$ , but has the same trace.

There is a nice analogy when looking at parametrized curves: consider a particle moving in space. Then the function describing the position of the particle at time  $t$  is a parametrized curve.

**Definition 1.3.** Let  $\alpha(t) = (x_1(t), x_2(t), \dots, x_n(t))$ , for  $t \in I$ . Then  $\alpha$  is called a *regular parametrized curve* if

$$|\alpha'(t)| \neq 0, \forall t \in I, \text{ where } \alpha'(t) = (x'_1(t), x'_2(t), \dots, x'_n(t))$$

The *unit tangent* of the curve  $\alpha$  is given by  $\vec{T} = \frac{\alpha'(t)}{|\alpha'(t)|}$ .

In the context of the moving particle analogy given above, the derivative of the curve gives the velocity of the particle at time  $t$  (a vector). Thus, a curve being regular means that the travelling particle never stops on his way from one end point to the other.

**Definition 1.4.** The euclidean length of a curve between  $t = a$  and  $t = b$  is defined as

$$s(a, b) = \int_a^b |\alpha'(t)| dt$$

A curve  $\alpha(t)$  is said to be in *arc length parametrization* if  $|\alpha'(t)| = 1 \forall t$  (i.e. any unit interval in  $\mathbb{R}$  map to an arc of unit length in  $\mathbb{R}^n$  by the function  $\alpha$ ).

Now, let  $t = s$  be an arc-length parametrization for the two-dimensional curve  $\alpha$ . Then,  $\vec{T}(s) = \alpha'(t)$ , since  $|\alpha'(t)| = 1$ . By differentiating both sides, we get

$$\begin{aligned} \kappa(s)\vec{N}(s) &= \alpha''(t) \\ \Rightarrow \kappa(s) &= |\alpha''(t)| \end{aligned}$$

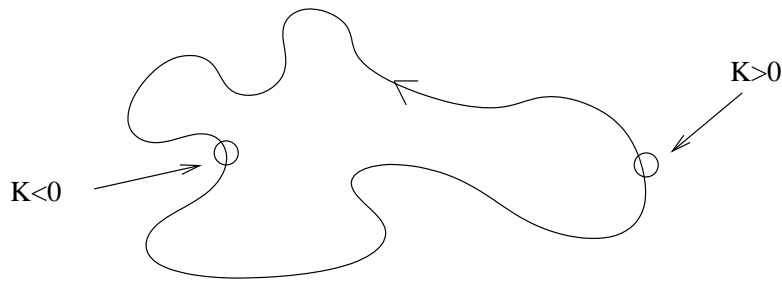


FIGURE 1.1. Sign of the curvature

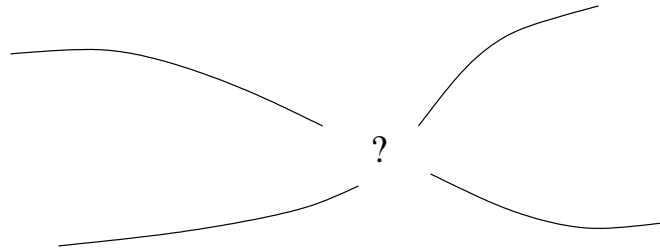


FIGURE 2.1. Example of good continuation

where  $\kappa(s)$  is a scalar, and is a measure of how much the curve is bending at a point. We call  $\kappa$  the *euclidean curvature* of  $\alpha$ . If  $\alpha$  is an open curve (the two endpoints are different), then we can give an arbitrary sign to the curvature. On the other hand, if  $\alpha$  is a Jordan curve (i.e. a closed curve that defines unambiguously a bounded region, called the interior, and an unbounded region, called the exterior), then we adopt the following convention for the sign of  $\kappa$ : it is positive if the point is on a “convex” part of the curve with respect to its interior, and it is negative if the point is on “concave” part, as shown in figure 1.1. The term curvature refers to two-dimensional curves; the analog definition in three dimensions for surfaces is called *torsion*. For the next few definitions, we are considering a smooth (differentiable) surface in three-dimensional space.

**Definition 1.5.** At any point on the surface, we can find 2 orthogonal vectors such that locally the surface is bending most in one direction and the least in the other direction. These 2 vectors are called the *principal directions*. The curvatures  $\kappa_1$  and  $\kappa_2$  of the surface along these directions are called the *principal curvatures*.

For example, the two principal curvatures of a sphere at any point are the same and are non-negative, since the surface is everywhere convex. The principle curvatures of a plane are both 0, since a plane is not bending in any direction.

**Definition 1.6.** The *mean curvature* is simply defined as  $H = \frac{(\kappa_1 + \kappa_2)}{2}$  and the *Gaussian curvature* as  $G = \kappa_1 \kappa_2$ .

## 2. BIEDERMAN’S PRINCIPLES OF ORGANIZATION

Before describing the principles of organization enumerated by Biederman at the end of the 80’s, we first briefly list the famous laws of grouping brought by the Gestalt group of psychologists, since these ideas are related to those of Biederman.

### 2.1. Laws of grouping (Gestalt).

- The good continuation (smoothness). In figure 2.1, how would you join the 4 line pieces ?

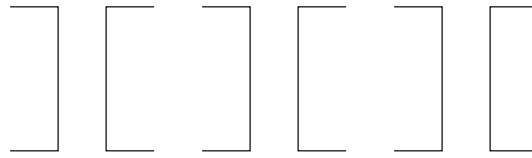


FIGURE 2.2. Example of closure

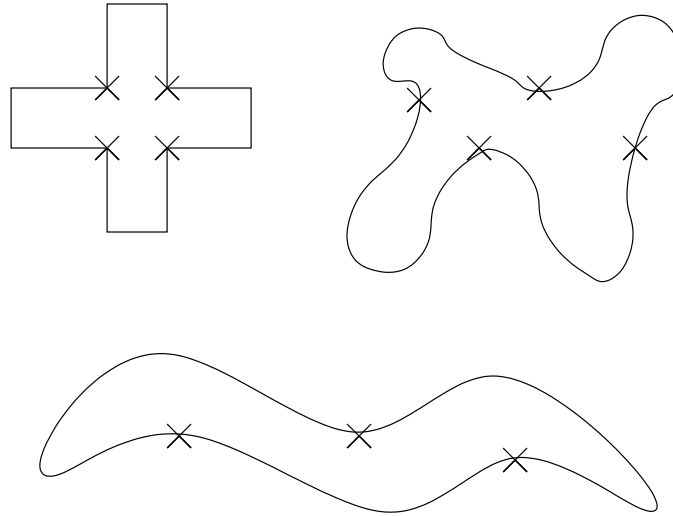


FIGURE 2.3. Segmentation at deep concavities

- Closure of shapes: we tend to group together sets of segments that, when joined, form closed curves. For example, in figure 2.2, do you see a row of adjacent boxes or pillars ?
- Common fate: an example of this is that we often group points moving in the same direction.
- Similarity: sets of points of the same colour for example.
- Proximity: we have the intuition that 2 parts that are close from each other are more likely to be related than two distant parts.

## 2.2. Principles of organization.

In 1987, Irving Biederman published in the *Psychological Review* a paper titled “Recognition-by-Components: A Theory of Human Image Understanding”, which exposes a theory on how humans reason about 2D images and recognize objects in a scene. Here is a quick summary of the main ideas of Biederman.

1. The object recognition process should not depend on precise, quantitative measurements on the images, since these are likely to be extremely sensitive on various parameters, like the illumination of the scene and the position of the camera for example.
2. Observe that we tend to segment shape contours in the plane at the regions of deep concavities (large negative curvature) to identify components. Figure 2.3 shows two examples of this principle, and one case where this rule does not agree with our perception. This hypothesis leads to the decomposition of a shape into a small number of components.
3. These components, called *geons*, come in a small number of types. Biedermann claims that  $\leq 36$  types of geons suffice so that their combinations lead to a high representational power (he uses only generalized cones).

4. Non-accidental properties of the model: curvature, collinearity, parallelism, symmetry and cotermination. The detection of these properties in an image is not sensitive to the viewing direction of the camera, lighting, image quality, etc. Remark the connection with the Gestalt laws of visual grouping.
5. Another aspect of the theory of recognition-by-component (RBC) is that when a small number of components or geons can be identified in an image, the objects in the scene can still be recognized efficiently even when some of the objects are partially occluded.

### 3. APPEARANCE-BASED RECOGNITION AND EIGENSPACE REPRESENTATION

#### 3.1. General idea.

In the previous section, we have summarized Biederman's view about object recognition. However, we have not described yet any method or algorithm that can be used in practice for recognition. In this section, we shall remedy to this situation, but only by a brute force approach.

Here is how we procede:

1. Build a large database of images collected in some controlled way. For example, we could fix an object to a turning plate, and, with a fixed camera, taking pictures at a regular rate.
2. Collect images for several (a lot of) objects.
3. To recognize an object from an image, query the database to find the closest matching object.

To have a hope that this method will work reasonably well, we need to make several assumptions:

- All the images are of a single object at a time. This way, we evitate the problem of partially occluded objects.
- No complicated background: without this assumption, the object could lie on a background that has features very similar to the object itself, making the distinction between foreground and background difficult.
- All images are of the same size and same intensity range. For the second requirement, we can simply normalize the intensities of all the images.
- We assume that we have a LOTS of images !
- The camera is fixed in the scene. In other words, the camera is looking at the object in the same direction and orientation for all the pictures taken.

Since we need to make a large number of comparisons between images, this approach certainly has efficiency issues. However, we can make the method feasible by a nice and useful trick: the eigenspace representation. This method is used for face recognition systems. Now, let's introduce formally the eigenspace representation.

#### 3.2. Eigenspace representation.

Let  $\vec{X}_1, \vec{X}_2, \dots, \vec{X}_n$  be  $n$  vectors of dimension  $N^2$  each (in this discussion, the term vector will denote vertical vectors). In our context, these vectors will represent square images ( $N \times N$ ) where the rows have been appended one after the other to form a long vector. However, keep in mind that this theory works for a general set of vectors. Let  $\tilde{X} = \frac{1}{n} \sum_{i=1}^n \vec{X}_i$  be the average of our set of  $n$  vectors. We can always express any vector  $\vec{X}_j$  by a sum of the average  $\tilde{X}$  with a deviation term:

$$\vec{X}_j = \tilde{X} + \sum_{i=1}^n g_{ji} \vec{e}_i \quad (4.1)$$

Let  $X$  be the  $N^2 \times n$  matrix defined as  $X = [(\vec{X}_1 - \tilde{X}) \dots (\vec{X}_n - \tilde{X})]$ . Define  $Q = XX^T$ , which is an  $N^2 \times N^2$  matrix. Any matrix of this form (some matrix times its transpose) is called a *covariance* matrix. The vectors  $\vec{e}_1, \vec{e}_2, \dots, \vec{e}_n$  in the above formula are the *eigenvectors* of  $Q$  (i.e. they satisfy  $Q\vec{e}_i = \lambda_i \vec{e}_i$  for some  $\lambda_i \in \mathbb{C}$ ; the  $\lambda_i$  are called the *eigenvalues* of  $Q$ ). Finally, the scalar  $g_{ji}$  is the projection of  $\vec{X}_j$  onto the eigenvector  $\vec{e}_i$  (i.e.  $g_{ji} = \langle \vec{X}_j, \vec{e}_i \rangle = \vec{X}_j^T \vec{e}_i$ ). The representation given by the formula (4.1) is exact and does not reduce the complexity. However, we can remark that the major components in the sum are the

terms where the eigenvectors correspond to the largest eigenvalues of  $Q$ . The so-called *principle components* are obtained by sorting the eigenvalues by decreasing magnitudes:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n > 0$$

We know that all the eigenvalues are strictly positive since the matrix  $Q$  is positive definite. Thus, the idea of the eigenspace representation is to approximate the vectors by  $\bar{X}_j \approx \tilde{X} + \sum_{i=1}^k g_{ji} \bar{e}_i$ , for some  $k < n$ . In other words, we restrict our space to the  $k$  most significant eigenvectors, and thus reducing the complexity. Based on this idea, we can now describe more precisely the appearance-based approach to object recognition.

### 3.3. Appearance-based algorithm.

- Learning phase (building our database)

1. We have a set of images of the same size. The intensities are normalized.

$$\bar{X} = \frac{[x_1, x_2, \dots, x_{N^2}]}{\sum_{i=1}^{N^2} x_i}$$

2. Say we have  $R \times L \times P$  images in total, where:
  - $R$  is the number of different view points we are using for each objects
  - $L$  is the number of different light source directions
  - $P$  is the number of objects
3. The *universal image set*  $U$  of all the objects is given by

$$\left\{ \bar{X}_{1,1}^{\rightarrow(1)}, \dots, \bar{X}_{R,L}^{\rightarrow(1)}, \bar{X}_{1,1}^{\rightarrow(2)}, \dots, \bar{X}_{R,L}^{\rightarrow(2)}, \dots, \bar{X}_{1,1}^{\rightarrow(P)}, \dots, \bar{X}_{R,L}^{\rightarrow(P)} \right\}$$

where  $\bar{X}_{r,l}^{\rightarrow(p)}$  is the image of the  $p^{th}$  object with the  $r^{th}$  view direction and the  $l^{th}$  light source direction.

4. Subtract the average vector for each image in  $U$ .

$$\bar{X} = \left[ \left( \bar{X}_{1,1}^{\rightarrow(1)} - \tilde{X} \right), \dots, \left( \bar{X}_{R,L}^{\rightarrow(P)} - \tilde{X} \right) \right]$$

5. Compute the  $N^2 \times N^2$  matrix  $Q = X X^T$ . Compute the first  $k$  eigenvectors of  $Q$ , corresponding to the largest  $k$  eigenvalues. The set  $\{\bar{e}_i | i = 1, \dots, k\}$  is called the *universal eigenspace*.
6. Appearance representation is then obtained by projecting each image from the training sequence (for a fixed object) onto the universal eigenspace:

$$g_{r,l}^{(p)} = [\bar{e}_1, \bar{e}_2, \dots, \bar{e}_k]^T \left( \bar{X}_{r,l}^{\rightarrow(p)} - \tilde{X} \right)$$

and this gives  $k \times 1$  vectors. In the case  $k = 3$ , we get a set of points in three dimensions. As  $R$  and  $L$  grow to  $\infty$ , the set of points  $\left\{ g_{r,l}^{(p)} | r = 1, \dots, R \text{ and } l = 1, \dots, L \right\}$  will form a continuous curve since small changes in the illumination direction or of the view point induce small changes in the  $g$ 's. Thus, we can interpolate these points to find one curve in  $k$ -dimensional space for each object (see figure 3.1).

- Recognition phase

1. Given a new image, we project it onto our universal eigenspace, as we did when building our database (step 6 above). As a result we obtain a  $k$ -tuple  $g$ .
2. Find the object in the database such that its corresponding curve in the eigenspace is the closest to the point  $g$ .
3. Now that we have identified an object, we can project our image onto the matching object's eigenspace only, using the same method, to find a point on the curve which is closest to  $g$ . This way we can estimate the viewing direction and light source direction parameters of our image.

A couple of remarks about this appearance-based recognition algorithm:

- This technique is very sensitive to the illumination of the scene and to occlusion of objects.

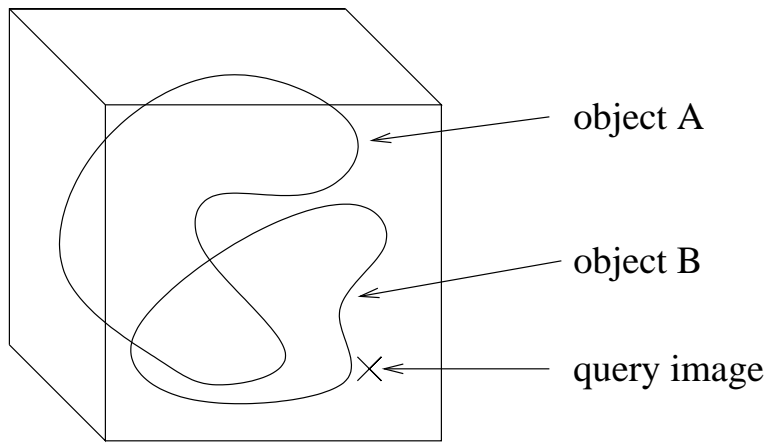


FIGURE 3.1. Curve interpolation for  $g_{r,l}^{(p)}$ , each closed curve corresponds to a fixed object ( $k = 3$ ).

- Adding new objects to the database is difficult. We have to recompute the average  $\tilde{X}$ , the matrix  $Q$  and its eigenvectors/eigenvalues.
- This method matches specific instances of objects. It cannot identify object that are of the same type or class.