Slope and Curvature Density Functions

The **slope density function** is the histogram of all slopes (tangent angles) in a contour. This function can be used for recognizing objects. We can also use the derivative of the slope representation, which we can call the **curvature representation**. Its histogram is the **curvature density function**. The curvature density function can also be used to recognize objects. Its advantage is its **rotation invariance**, i.e., matching two curvature density functions determines similarity even if the two objects have different orientations.

---

Examples of Slope Density Functions

---

Signature

Another popular method of representing shape is called the **signature**. In order to compute the signature of a contour, we choose an (arbitrary) **starting point A** on it. Then we measure the **shortest distance** from A to any other point on the contour, measured in **perpendicular direction** to the tangent at point A. Now we keep moving along the contour while measuring this distance. The resulting measure of distance as a function of **path length** is the signature of the contour.

---

Signature

---

Bending Energy

---

Region Skeleton
Region Skeleton

![Region Skeleton](image)

Fig. 7.15 Templates used by the thinning algorithm. (a) Thinning templates; (b) Restoring templates. A pixel is removed if we center the templates on it and find a match for any thinning template and no match for any restoring template.

April 3, 2018 Computer Vision
Lecture 14: Shape Representation II

Region Skeleton

![Region Skeleton](image)

Fig. 7.16 Result of applying the thinning algorithm to various shapes. (a) Input image; (b) Computed skeletons superimposed on gray copies of the original shapes.

April 3, 2018 Computer Vision
Lecture 14: Shape Representation II

Fourier Transform of Boundaries

As we discussed before, a 2D contour can be represented by two 1D functions of a parameter s. Let S be our (arbitrary) starting point on the contour. Let us move in counterclockwise direction (for example, we can use a boundary-following algorithm to do that).

In our discrete (pixel) space, the points of the contour can then be specified by \(v[s], h[s]\), where s is the distance from S (measured along the contour) and v and h are the functions describing our vertical and horizontal position on the contour for a given s.

April 3, 2018 Computer Vision
Lecture 14: Shape Representation II

Fourier Transform of Boundaries

![Fourier Transform of Boundaries](image)

Obviously, both \(v[s]\) and \(h[s]\) are periodic functions, as we will pass the same points over and over again in constant time intervals.

This gives us a brilliant idea: Why not represent the contour in Fourier space?

In fact, this can be done. We simply compute separate 1D Fourier transforms for \(v[s]\) and \(h[s]\).

When we convert the result into the magnitude/phase representation, the transform gives us the functions \(M_f[l], M_h[l], \phi_v[l], \phi_h[l]\) for frequency \(l\) ranging from 0 (offset) to \(s/2\).

April 3, 2018 Computer Vision
Lecture 14: Shape Representation II
Fourier Transform of Boundaries

As with images, higher frequencies do not usually contribute much to a function and can be disregarded without any noticeable consequences. Let us see what happens when we delete all functions above frequency $l_{\text{max}}$ and perform the inverse Fourier transform. Will the original contour be conserved even for small values of $l_{\text{max}}$?