Shadow Image Recognition

Abstract

This article describes a technique for recognition and classification of 2D images.

1 Problem

Given a 2D black and white key image and a set of 2D black and white images, we want to define a process that can rank the images from the given set in terms of their resemblance to the key image. We require that images are reasonably centered and oriented but the scale factors are arbitrary.

For our purposes we use a square as the key image that we want other images to be ranked against.

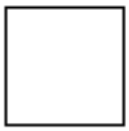
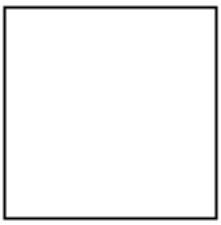


Figure 1: Key image

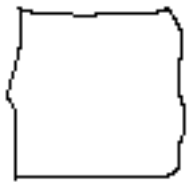
Our test set consists of the following images.

- Larger square

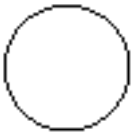


- Smaller square

- Unperfect square



- Circle



- Triangle

- Vertical line

We want to rank images from the given set against the key square. Ideally, the two squares from the test set would have a higher rank and the circle and triangle lower ones. Perhaps circles should have a higher rank than triangles as they seem to be closer to squares intuitively.

2 Solution

Our solution is based on what we call shadow images that are automatically generated from the original black and white images. Our shadow images are grayscale. For example the generated shadow image for the key square looks like the following.

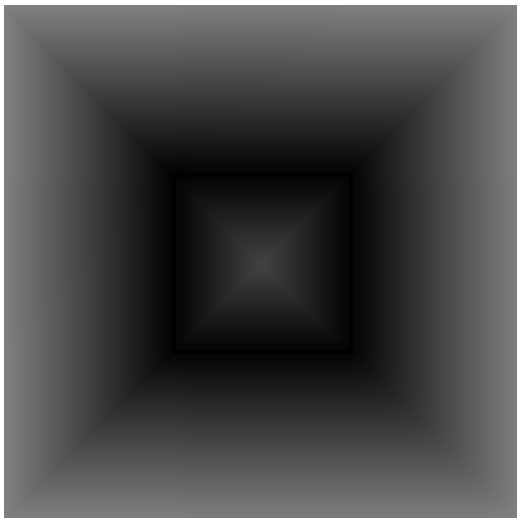


Figure 2: Key shadow image

The most important characteristic of the presented method is that by generating shadow images we distribute the features of the original shape across the whole 2D field. The generated shadow images for the circle and vertical line are:

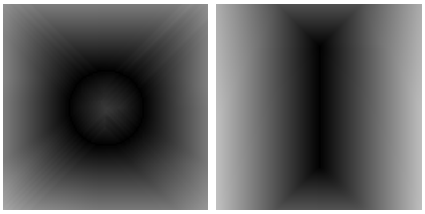


Figure 3: Shadows of the circle and vertical line

In essence, the color gradually becomes lighter as we get farther from the black pixels of the original image. On the shadow images, the pixel values are in range [0, 255] while on the original images the background pixels have value, 255 and foreground ones, 0.

Next, we take what we call shadow regions of the key and test images. Each region represents a path of a certain width around the center of the image like it is shown on the picture below.

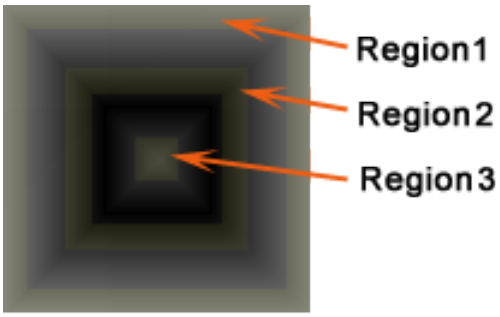


Figure 4: Key image shadow slicing

We pick the same exact slices for test images:

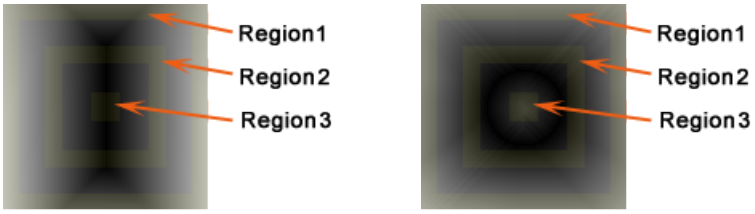


Figure 5: Test images shadow slicing

The first picture is the vertical line and the second one is the circle. Now we will calculate diff vectors between regions of the key and test shadows.

Listing 1: Pseudocode for calculating region differences

```
key_shadow = generate_shadow_image(key_square);
test_shadow = generate_shadow_image(circle);

region = Region1;
vector diff;
```

```
foreach(position in region) {
  pixel_diff = key_shadow.get_pixel(position) -
     test_shadow.get_pixel(position);
  diff.push_back(pixel_diff);
}

return diff;
```

We calculate diff vectors, one per shadow region for all pairs of the key and test images. To rank a test image, we calculate the standard deviations of the diff vectors. Since we use three shadow regions, there will be three diff vectors per key/test image pair. It means that for each such pair we'll get three standard deviations (one deviation per shadow region).

2.1 Test Results

For testing, we used 128x128 images. The key and test images are described in the first section. The shadow region width was 12 pixels, with three regions. The left top coordinates of the regions: Region1-(0,0), Region2-(30,30) and Region3-(52,52). The results are shown in the table below.

Test Image	Region1 Deviation	Region2	Region3	Final Rank
Larger square	1.6	20.5	1.5	1.55
Smaller square	2.0	2.3	3.2	2.15
Unperfect square	3.5	16.3	3.4	3.5
Circle	8.9	7.5	3.1	5.3
Triangle	18.4	18.9	11.6	15
Vertical line	44.1	27.6	16.9	22.25

The final rank in the last column is calculated as an average of two two smallest deviations. For example, for the circle:

$$Rank = (7.5 + 3.1)/2 \tag{1}$$

We discard the largest deviation because we assume for the possibility of various scales of the key and test images. When the internal and external features of shadow images intersect due to some difference in scales, the shadow regions where it happens will have high standard deviations. We try to ignore such regions.

Again the smaller deviations (final rank) indicate a better match. The table shows that the program correctly ranked the test images with the three squares being at the top followed by the circle and with the triangle and vertical line far at the bottom.

3 Conclusion

The process can be summarized:

- For the key and test images, generate corresponding shadow images.
- Pick a set of shadow regions identical for all shadow images.

- Calculate a diff vector between the shadow regions of the key and test images.
- Calculate the standard deviation for each of the diff vectors.
- Smaller deviations indicate better matches.

A few words about shadow regions. We used three regions to be able to match similar images of different scales properly such as the smaller and larger squares. Choosing the number of regions and their width is somewhat of an art. If images can be identified by their external shape only then a single most outer region would be enough because the remote parts of the shadow carry the external features of the original image. When it is known that the relative scales of key and test images are the same then one region covering the whole image should work fine.

The shadow image technique could also be used for finding relative image scales and orientations. In machine learning, they could represent training sets. In turn machine learning could be used to optimize the shadow region selection for example.

List of Figures