Accurately and Efficiently Identifying Centers of Circular Symmetry in Images

Siddharth Trehan

Electrical Engineering and Computer Science Student, MIT

Abstract

In cases such as the detection of circles at any scale in an image, it is important to be able to quantify the degree of circular symmetry about a point and use it to identify centers of high degree of circular symmetry in an image. This writeup describes a good metric for quantifying amount of circular symmetry and an efficient algorithm for finding the degree of rotational symmetry about every point in an $N \times M$ image at once in $O(NM \lg NM)$ time using gradient and cross-correlation.

Keywords: Image Processing, Machine Vision, Cancer Cell Detection

1. Overview

Circular symmetry is defined by the property of an image U and a point P such that all points of the same distance from P will have the same brightness – that is, pixel brightness is only a function of distance from P. Images that are perfectly rotationally symmetric also have the property that rotation by any angle about P does not change the image. Many images, however, are not perfectly circularly symmetric about a point but have some degree in which rotation preserves the original image. For these images, it would be useful to quantify the degree to which circular symmetry is present about a particular point.

To develop a good metric, we note another property of circularly symmetry: images with perfectly circular symmetry have the property that the brightness gradient vector field of the image will be oriented so that each gradient vector points either towards or away from the center of symmetry. A good way to quantify the degree of circular symmetry would then be to take the sum of squares of the tangential projection of each gradient vector relative to the center of symmetry, which should be zero in the case of perfect circular symmetry. This particular metric will become convenient in the techniques used in this writeup, and we claim it will accurately identify centers of circular symmetry, though we will identify flaws in it in the last section.

The technique we use to quantify the degree of circular symmetry about every point in the image efficiently will be to encode components of the gradient vector as different pixels in the image and cross-correlate it with a special image (which we will call the template) that computes this sum of squares of tangential projections. This algorithm will run in $O(NM \lg NM)$ time using the fast Fourier transform (FFT). We will also apply different filters to the image before convolution and investigate different transformations that can be applied to the template in order to increase accuracy.

2. Quantifying Circular Symmetry About A Point

Let $U(\vec{r})$ be the brightness of the pixel at position vector $\vec{r}$ in the image. Let $\vec{G}(\vec{r}) = \nabla U(\vec{r})$ be the gradient of U, which denotes the direction of maximum increase in image brightness at a particular pixel location (we will leave the discussion of which kernels are best suited for computing $\vec{G}$ for Section 5). Let's also suppose that we are trying to quantify the amount of circular symmetry about the point O located at position vector $\vec{o}$.

Consider a single brightness gradient vector located at point Q, $\vec{G}(\vec{q})$. As discussed in Section 1, if U is perfectly circularly symmetric about O, $\vec{G}(\vec{q})$ should not have any tangential component – that is, $\vec{G}(\vec{q}) \cdot \hat{t}_{\vec{q}-\vec{o}} = 0$, where $\vec{t}_{\vec{q}-\vec{o}}$ is the tangential vector, a normalized vector orthogonal to the radial vector $\vec{q} - \vec{o}$. Therefore, if we compute the tangential projections of $\vec{G}$ at every point in the image, the more circularly symmetric the image is, the closer the tangential projections will be to 0. For an aggregate measure of circular symmetry about a point, we might simply add together the tangential projections. However, because $\vec{G}$ is a conservative vector field, adding together the tangential projections will only give a result of zero. Therefore, we will square the tangential projection at each pixel and add them together. We can then quantify the error associated with the degree of circular symmetry as:

$$\epsilon = \sum_{\vec{q}} (\vec{G}(\vec{q}) \cdot \hat{t}_{\vec{q}-\vec{o}})^2 \tag{1}$$

Though it will not be the focus of this writeup, we can also quantify the error associated with the degree of elliptical symmetry by adjusting $\hat{t}_{\vec{q}-\vec{o}}$ to point in a direction tangential on an ellipse instead of simply orthonormal to the radial vector. Symmetries associated with other shapes, such as rectangles and triangles, can be found similarly as well.

3. Computing Circular Symmetry About A Point

If $\vec{G}(\vec{q})$ and $\hat{t}_{\vec{q}-\vec{o}}$ in Equation 1 were scalars and if it were just a sum instead of sum of squares, we could separate out a vector of the image brightness gradients and a vector of the tangential vectors and compute the dot product of the two vectors as a way to get ϵ. Formulating the problem as the dot product between the image gradient and some sort of template is convenient because it allows us to efficiently compute the circular symmetry about *every* point efficiently using cross-correlation as the next step (the topic of Section 4). Regardless, we can still easily formulate the problem as the dot product, though we will have to adjust the image so its pixels can accommodate vector components and not just scalars.

First, let's focus only on the contribution of a single pixel $\vec{q}$ to the error ϵ, and let's define G_x and G_y as the components of $\vec{G}(\vec{q})$ and t_x and t_y as the components of $\hat{t}_{\vec{q}-\vec{o}}$. Then:

$$\begin{aligned}
\epsilon_{\vec{q}} &= (G_x t_x + G_y t_y)^2 \\
&= (G_x^2)(t_x^2) + 2(G_x G_y)(t_x t_y) + (G_y^2)(t_y^2) \\
&= (G_x G_x)(t_x t_x) + (G_x G_y)(t_x t_y) + (G_x G_y)(t_x t_y) + (G_y G_y)(t_y t_y) \\
&= \begin{pmatrix} G_x G_x \\ G_x G_y \\ G_y G_x \\ G_y G_y \end{pmatrix} \cdot \begin{pmatrix} t_x t_x \\ t_x t_y \\ t_y t_x \\ t_y t_y \end{pmatrix}
\end{aligned} \tag{2}$$

With Equation 2 in mind, we apply the following transformations to the image and the template:

$$\boxed{U(\vec{q})} \Rightarrow \vec{G}(\vec{q}) \Rightarrow \begin{array}{|c|c|}\hline G_xG_x & G_yG_x \\ \hline G_xG_y & G_yG_y \\ \hline\end{array}$$

Figure 1: Transformation of the pixels in the original image to produce the gradient components image.

$$\vec{q} - \vec{o} \Rightarrow \hat{t} \Rightarrow \begin{array}{|c|c|}\hline t_xt_x & t_yt_x \\ \hline t_xt_y & t_yt_y \\ \hline\end{array}$$

Figure 2: Construction of the template containing components of the tangential vector.

1. For each pixel in the image, compute the components of the gradient G_x and G_y for that pixel and replace that pixel with four new pixels that have the values G_xG_x, G_xG_y, G_yG_x, and G_yG_y (as shown in Figure 1). We will call the resulting image the gradient components image.
2. Construct the template by first computing $\hat{t}$ at each pixel location, and find the components t_x and t_y. Then, for each position, create four pixels arranged exactly as before with the values t_xt_x, t_xt_y, t_yt_x, and t_yt_y (as shown in Figure 2).

Now, dot product between the template image and the gradient components image will be equal to ϵ as defined in Equation 1 and quantifies the error associated with circular symmetry – that is, the larger the dot product is, the less circular symmetry is present about point $\vec{o}$.

4. Identifying Centers of Circular Symmetry

Identifying centers of circular symmetry requires recomputing the dot product for each value of $\vec{o}$ and choosing the best. This can be accomplished by constructing a template for $\vec{o} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and computing the cross-correlation between the gradient components image and the template, which we will call the circular symmetry error matrix (CSEM). The entry at $\vec{r}$ in this matrix corresponds to ϵ at $\vec{o} = \vec{r}$. Using this 2-D FFT, this can be computed in $O(NM \lg NM)$ time.

Using the CSEM, we can identify good centers of circular symmetry by a combination of thresholding and peak finding.

5. Further Optimizations for Accuracy

The first question that arises when implementing the algorithm is how to choose a proper kernel to compute the brightness gradient. To show how choosing the right kernel is important, let's see what happens if we choose the simple kernel that subtracts the current pixel from the next one.

Suppose we are given an arbitrary image and we are using this simple gradient kernel. Let's focus only on a thin circular ring of any radius around $\vec{o}$ with a thickness of 1 pixel and examine its contribution to ϵ. Assume that this ring consists of n pixels, of brightness E_0, E_1, ..., E_{n-1}, that are arranged end-to-end, and that the brightness gradient at pixel i in the tangential direction is equal to $E_{i+1 \bmod n} - E_i$, which is a good approximation to the value that the aforementioned gradient kernel would compute. Then, the contribution of the pixels in this ring to ϵ is:

$$
\begin{aligned}
\epsilon_r &= \sum_{i=0}^{n-1} (E_{i+1 \bmod n} - E_i)^2 \\
&= \sum_{i=0}^{n-1} (E_{i+1 \bmod n}^2 + E_i^2 - 2E_{i+1 \bmod n} E_i) \\
&= \sum_{i=0}^{n-1} E_{i+1 \bmod n}^2 + \sum_{i=0}^{n-1} E_i^2 - 2\sum_{i=0}^{n-1} E_{i+1 \bmod n} E_i \\
&= 2\sum_{i=0}^{n-1} E_i^2 - 2\sum_{i=0}^{n-1} E_{i+1 \bmod n} E_i \\
&= 2n \left(\mathrm{Exp}[E_i^2] - \mathrm{Exp}[E_{i+1 \bmod n} E_i]\right) \\
&= 2n \left(\mathrm{Exp}[E_i^2] - \mathrm{Exp}[E_i]^2 + \mathrm{Exp}[E_i]^2 - \mathrm{Exp}[E_{i+1 \bmod n} E_i]\right) \\
&= 2n \left(\mathrm{Var}[E_i] - (\mathrm{Exp}[E_{i+1 \bmod n} E_i] - \mathrm{Exp}[E_i]^2)\right) \\
&= 2n \left(\mathrm{Var}[E_i] - (\mathrm{Exp}[E_{i+1 \bmod n} E_i] - \mathrm{Exp}[E_{i+1 \bmod n}]\mathrm{Exp}[E_i])\right)
\end{aligned}
\tag{3}
$$

With the final result:

$$
\epsilon_r = 2n \left(\mathrm{Var}[E_i] - \mathrm{Cov}[E_{i+1 \bmod n}, E_i]\right) \tag{4}
$$

Where Exp, Var, and Cov are the expectation, variance, and covariance operators, respectively. This result shows us three things:

1. ϵ weighs rings with more pixels (that are further away) more heavily than those that are closer to the center of symmetry. This can be a problem because the degree of circular symmetry tends to dissipate with distance from the center of symmetry. However, this problem can be easily overcome by adjusting the definition of the $\hat{t}$ vectors so that they no longer have to be normalized, and their magnitude decreases with distance from the center of symmetry. In fact, different weights can be given to different individual pixels by adjusting the magnitude of the corresponding $\vec{t}$ vector. This can also be used to search for circles with radii within a specific range by setting all $\vec{t}$ vectors to 0 except for those within that distance range from the center of symmetry.
2. ϵ increases with the variance of pixel brightness in the circular rings. This is a property that agrees with our intuitive idea of circular symmetry – that the more similar in brightness pixels of a given radius are to one another, the more circular symmetry the image has.
3. ϵ increases as the covariance factor in Equation 4 decreases. This covariance term can be understood as the degree to which rotation of this circular ring by a tiny amount preserves the original ring. While this also agrees with our intuitive understanding of circular symmetry, it limits the quantification of circular symmetry because it only takes into account degree of matching if a tiny amount of rotation is applied, and not a large range of rotation amounts. A ring consisting of alternating black and white pixels, for example, would have a relatively high degree of circular symmetry according to intuition, but it has a covariance of 0 and therefore a very high ϵ. This is the issue that arises when using image gradient kernels that only look at a small number of neighboring pixels. Therefore, it is best to use a large kernel, such as a Gaussian gradient kernel with a large σ.