The Angular Difference Function and its Application to Symmetry Detection

Yosi Keller*, Yoel Shkolnisky† and Amir Averbuch‡

Abstract

We present an algorithm for detecting cyclic and dihedral symmetries of an object. Both symmetry types can be detected by the special patterns they induce on the object's Fourier transform. These patterns are effectively detected and analyzed using the "angular difference function" (ADF), which measures the difference between images in the angular direction. The ADF is accurately computed by using the pseudo-polar Fourier transform, which rapidly computes the Fourier transform of an object on a near-polar grid. The proposed algorithm detects all axes of centered and non-centered symmetries. It is algebraically accurate and uses no interpolations.

1 Introduction

The two most common types of symmetries are rotational and reflectional symmetries. An object is said to have a rotational symmetry of order N if it is invariant under rotations of $\frac{2\pi}{N}n$, $n = 0 \dots N-1$. An object is said to have a reflectional symmetry if it is invariant under a reflection transformation about some line. Most existing algorithms usually detect either rotational or reflectional symmetry. The algorithm presented in this paper is based on the angular difference function (ADF), which measures the difference between two objects in a given angular direction. For symmetric objects the value of this function is shown to be zero in points that correspond to the symmetry axes. The zeros of the ADF identify both rotational and reflectional symmetries.

*yosi.keller@yale.edu
†yoel@math.tau.ac.il
‡amir@math.tau.ac.il

The algorithm characterizes rotational symmetries by the set of rotation angles that keep the object unchanged. Similarly, it characterizes reflectional symmetries by the set of reflection axes.

The idea behind the proposed algorithm is related to the work presented in [1]. Both algorithms detect the patterns that symmetries induce in the frequency domain. However, the algorithm we present in this paper uses an algebraically exact method for detecting these patterns. Specifically, it computes the ADF using the pseudo-polar Fourier transform and then uses the zeros of the ADF to detect minima ridges in the Fourier domain. It also uses a simpler scheme to infer the reflectional symmetry from the rotational symmetry.

The paper is organized as follows. In section 2 we describe previous work related to symmetry detection. In section 3 we mathmatically define rotational and reflectional symmetries and describe the relations between both symmetry types. In section 4 we describe the pseudo-polar Fourier transform, which evaluates the Fourier transform of an object on a near-polar grid. This transform is the basis for our symmetry detection algorithm. In section 5 we introduce the Angular Difference Function (ADF) as a tool for analyzing polar properties of images and utilize it to detect and analyze rotational and reflectional symmetries. In sections 7 and 8 we present experimental results and some concluding remarks.

2 Previous work

Symmetry is thoroughly studied in the literature from both theoretical, algorithmic and applicative perspectives. Theoretical treatment of symmetry can be found in [2, 3]. The algorithmic approach to symmetry detection can be divided into several categories based on its characteristics. The first characteristic of a symmetry detection algorithm is whether it considers symmetry as a binary or continuous feature, which measures the amount of symmetry. A second characteristic is the type of symmetry detected by the algorithm. Most algorithms detect either rotational or reflectional symmetry but not both. A third characteristic is the assumptions on the image. For example, whether the algorithm assumes that the image is symmetric or detects it itself, or whether the algorithm assumes that the symmetric feature is located at the center of the image. A Fourth characteristic is whether the algorithm operates in the image domain or transforms the problem into a different domain, like the Fourier domain. A fifth characteristic is the robustness of the algorithm to noise and its ability to operate on real-life non-synthetic images. The last characteristic of an algorithm is its complexity. This characteristic is important for symmetry detection algorithms since most algorithms typically require an exhaustive search over all potential symmetry axes. Such a search requires excessive computation even for small images.

In the light of these characteristics we will examine the existing work on symmetry detection. Some of the work we describe refers to 3D symmetry detection algorithms. We describe such algorithms if they are applicable to 2D problems.

[4] presents a low-level, context free operator for detecting points of interest within an image, which relies on the assumption that context free attention is directed by symmetry. The suggested symmetry operator constructs the symmetry map of the image by assigning symmetry magnitude and symmetry orientation to each pixel. This map is an edge map where the magnitude and orientation of each edge depend on the symmetry associated with each of its pixels. The proposed operator allows processing different symmetry scales, enabling it to be used in multi-resolution schemes. Generally, the transform iterates over all pixels in the image, and for each pixel p it inspects all pairs of points in a neighborhood with midpoint p and radius r. It then computes the contribution of each pair according to its gradient and distance from p. The symmetry value of a point p is obtained by summing all contributions of the individual pairs. The direction of the symmetry at a point p is obtained by averaging the directions of the pair with the highest symmetry contribution to p. The proposed operator is demonstrated to be effective in detecting points of interest in natural images.

[5] uses such a local symmetry operator to construct an algorithm for detecting areas with high local reflectional symmetry. It defines a 2D reflectional symmetry measure as a function of four parameters x, y, θ, and r, where x and y are the center of the examined area, r is its radius, and θ is the angle of the reflection axis. Since examining all possible values of x, y, r, and θ is computationally prohibitive, the algorithm formulates the problem as a global optimization problem and uses a probabilistic genetic algorithm to find the optimal solution.

As noted previously, symmetry can be considered as either a binary or continuous feature. [6] treats symmetry as a continuous feature and defines the symmetry distance to measure the amount of symmetry in an object. For an object, given by a sequence of points, the symmetry distance is defined as the minimum distance in which we need to move the points of the original object in order to obtain a symmetric object. This also defines the symmetry transform of an object as the symmetric object that is closest of the given one. The paper [6] describes algorithms for computing the symmetry transform of an object with respect to rotational and reflectional symmetries, and handles the problem of selecting points to represent 2D objects. The suggested algorithms require finding point correspondence, which is generally difficult, and perform exhaustive search over all potential symmetry axes, which is computationally expansive.

While the works [5, 6] operate in the image space, it may be useful to transform the problem into a different domain. [7] suggests a method for estimating the relative rotation of two patterns using the Zernike moments. This problem is closely related to the problem of detecting rotational symmetry in images. Given two patterns, where one pattern is a rotated version of the other pattern, the Zernike moments of the two images will have the same magnitude and some phase difference. The phase difference can be used to infer the relative rotation angle of the two images. Given two patterns, the algorithm computes several of their Zernike moments and uses these moments to construct a probability density function that describes the probability of each rotation angle. It then identifies the relative rotation of the two patterns as the angle with the highest probability.

Another transform approach is given by [8]. [8] describes an algorithm for computing a reflective symmetry descriptor that measures the amount of reflective symmetry in 3D volumes for all planes through the center of mass. The descriptor maps any 3D volume to the sphere, where each point on the sphere represents the amount of symmetry in the object with respect to the plane perpendicular to this point. Each point on the sphere represents integration over the entire volume and therefore the descriptor is insensitive to noise and to fine differences between objects. The algorithm is computationally intensive as it requires exhaustive search over all planes in the image space. Specifically, for an $N \times N \times N$ object it requires $O(N^4 \log N)$ to compute the symmetry descriptor at full resolution. The shape signature defined in [8] can detect only reflective symmetry, as opposed to the descriptor suggested in the current paper, which can be used to measure both reflective and rotational symmetry.

The work [9] suggests using extended Gaussian images to detect rotational and reflectional symmetry. For a given input image, the algorithm uses some tessellation to construct the corresponding Gaussian image and then finds the correlation of the Gaussian image with itself. The algorithm assumes that the orientation histogram has the same symmetry as the original image and uses correlation to find the direction of strongest symmetry. This algorithm is computationally intensive and depends on the resolution of the tessellation. Moreover, the suggested method does not determine if the image is symmetric or not but rather assumes that the object possesses some degree of symmetry. Also, it does not find all symmetry axes but only the most dominant symmetry in the object.

The concept of deriving a symmetry invariant transformation is studied also by [10]. [10]

develops a method for detecting local, global, and skewed symmetries by using an affine invariant representation. For each feature point, the algorithm constructs an affine invariant feature vector whose entries represent relatively affine invariant quantities. These feature vectors represent both local features, which are sensitive to noise, and more global smoothed information, which is less sensitive to noise. Using these feature vectors it constructs a similarity matrix, which encodes the property that rotationally and reflectionally symmetric points have the same feature vector. The algorithm detects these symmetries by detecting lines with slope +1/-1 in the similarity matrix. If the number of points on the line is equal to the number of points on the contour than the pattern is rotationally symmetric and the degree of rotational symmetry is given by the number of detected lines. For reflectional symmetry a further refinement is required in order to detect only significant symmetries. The algorithm suggested by [10] requires determining feature points on the contour of the input object.

[1] gives a Fourier based approach for detecting reflectional and rotational symmetries. Given an input image, the algorithm normalizes it and rotates it by some arbitrary angle. The algorithm then computes the difference between the FFT magnitudes of the original image and the rotated image. The zero crossings of this difference are shown to correspond to the symmetries of the input image. These zero crossings are detected using a generalization of the algorithm in [11]. Another similar transformation is then used to distinguish between reflectional and rotational symmetries.

For some applications it is sufficient to reduce the problem of symmetry detection to the problem of pattern detection. [12] describes an algorithm for detecting patterns in an image, where some of the patterns exhibit rotational symmetry. The algorithm constructs the orientation image from the input image, and then applies normalized convolutions to the orientation image using a special set of filters. Each filter corresponds to one of the searched patterns. The algorithm does not detect whether the image is rotationally symmetric as defined in section 1, but rather detects symmetric patterns, like circles, stars and spirals. Detecting each pattern requires a convolution with a different filter.

Some of the suggested algorithms detect symmetry under somewhat restrictive assumptions. [13] presents an algorithm for detecting vertical reflectional symmetry using a 1D odd-even decomposition. The algorithm assumes that the symmetry axis is vertical and thus scans each horizontal line in the image. Each such line is treated as a 1D signal, which is normalized and decomposed into odd and even parts. Using the odd and even parts the algorithm constructs a target function which achieves its maximum at the point of mirror symmetry of the 1D signal. When the image has a vertical symmetry axis, all symmetry points of the different horizontal lines lie along a vertical line in the image. To detect symmetry axes that are not vertical, the algorithm requires prior knowledge on their directions.

Another algorithm that uses assumptions on the input image is given by [14]. This work describes an algorithm for detecting global reflectional symmetries in images that contain dense arrangements of local features such as line segments. The suggested algorithm [14] is based on psychological experiments that show that human vision utilizes grouping for symmetry detection. The algorithm consists of three stages. The first stage identifies clusters in the image, where each cluster contains elements with sufficient mutual affinity. This stage is implemented using an iterative algorithm that gradually refines the probability that each line segment belongs to a cluster. The second stage processes only the clusters detected in the first stage and detects pairs of symmetrical clusters together with their symmetry axis. The third stage applies the Hough transform to the symmetry axes detected in the second stage to detect more global symmetries.

Segmentation and symmetry detection are closely related problems. [15] presents an algorithm for simultaneous segmentation and symmetry detection. By using heuristic criteria and experimentation, the authors derive a target function that measures the fitness of the image to an assumed symmetry and segmentation. The algorithm than constructs a graph, where each node in the graph corresponds to a pair of points in the image and the cost of each edge in the graph is the value of the target function. Each path in the graph corresponds to a possible segmentation. Dikstra's algorithm is then used to find the minimal path, which corresponds to the segmentation with minimal cost.

For completeness of the survey we will present some applications of symmetry detection. Symmetry detection is widely used for tasks like vision and recognition. The underlying assumption is that points of interest exhibit a high degree of symmetry, which can be used as a cue for segmentation and recognition [4]. For example, [16] uses this assumption for automatic detection of points of interest by detecting points of high radial symmetry. Another vision application is presented in [17] for automated license plate extraction. [18] presents a human identification algorithm using symmetries in the spatial-temporal domain. The algorithm uses a sequence of images describing gaits and tries to recognize the person by using the spatio-temporal symmetry map of the image sequence. Symmetry detection is also used for classification, where objects are classified into different groups based on their symmetry properties. Such an application is demonstrated in [19] for coarse classification of Chinese characters.

3 Types of symmetries

In this paper we consider two types of symmetries, namely, rotational and reflectional symmetries. We will follow the notation used in [1] and [3].

An image $\psi(x, y)$ is said to have rotational symmetry of order $N \in \mathbb{N}$ if

$$\psi = R(\beta_n)\psi \tag{3.1}$$

for each $\beta_n = \frac{2\pi}{N}n$, $n = 0, \ldots, N-1$, where $R(\beta)$ is the 2D rotation matrix given by

$$R(\beta) = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix}. \tag{3.2}$$

An image $\psi(x, y)$ is said to have reflectional symmetry with respect to the line $y = (\tan\alpha)x$ if

$$\psi = S(\alpha)\psi \tag{3.3}$$

where $S(\alpha)$ is the 2D reflection matrix given by

$$S(\alpha) = \begin{pmatrix} \cos 2\alpha & \sin 2\alpha \\ \sin 2\alpha & -\cos 2\alpha \end{pmatrix}. \tag{3.4}$$

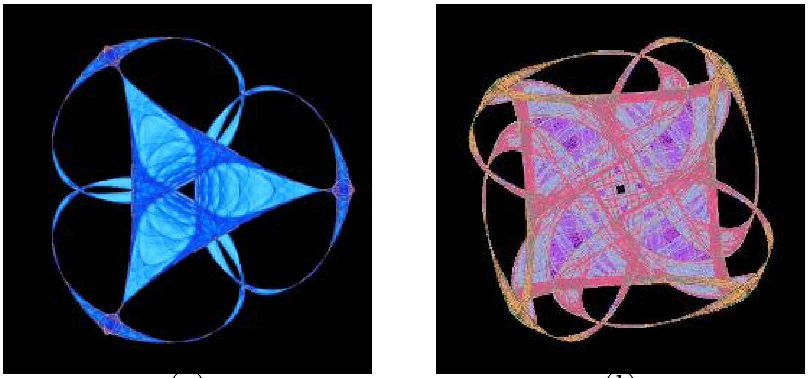
We say that an image ψ has reflectional symmetry of order N if there are N angles α_n that satisfy Eq. (3.4). If an image ψ has rotational symmetry of order N then it has reflectional symmetry of order N or has no reflectional symmetry at all ([3]). If an image has both rotational and reflectional symmetries then the angles of reflectional symmetry axes are given by

$$\alpha_n = \alpha_0 + \frac{1}{2}\beta_n \qquad n = 0, \ldots, N-1 \tag{3.5}$$

where α_0 is the angle of one of the reflectional symmetry axes and β_n are the angles of rotational symmetry.

An image ψ that has rotational symmetry of order N and no reflectional symmetry is said to have cyclic symmetry C_N (Fig. 1b). An image that is not rotationally symmetric is considered to have symmetry C_1. An image ψ that has both reflectional and rotational symmetries of order N is said to have dihedral symmetry D_N (Fig. 1a). As stated above, for any 2D image ψ, C_N and D_N are the only possible central symmetries [3].

Our algorithm takes an image ψ of dimensions $n \times n$ and computes the angles β_n of rotational symmetry and the angles α_n of the axes of reflectional symmetry, if such symmetry exists.



(a) (b)

Figure 1: Dihedral and rotational symmetry. (a) Dihedral symmetry D_3. (b) Rotational symmetry C_4.

4 The pseudo-polar Fourier transform

The proposed symmetry detection algorithm is based on the pseudo-polar Fourier transform [20]. The pseudo-polar Fourier transform evaluates the 2D DFT of a function on an oversampled set of angularly non-equispaced frequencies, which we call the pseudo-polar (PP) grid. Both the forward and inverse pseudo-polar Fourier transforms can be implemented using fast algorithms. Moreover, their implementation requires only 1D equispaced FFT's. In particular, the algorithm does not require re-gridding or interpolation. For a detailed description of the pseudo-polar Fourier transform see [20].

In section 4.1 we present the Fractional FFT. In section 4.2 we describe pseudo-polar Fourier transform and how to compute it using the Fractional FFT. In section 4.3 we conclude the presentation of the pseudo-polar Fourier transform by presenting its geometric interpretation.

4.1 Fractional FFT

The Fractional FFT (FRFT) [21], with its generalization given by the Chirp Fourier transform [22], is a fast $O(N \log N)$ algorithm that evaluates the DFT of a function on any equally spaced set of N points on the unit circle. Specifically, the FRFT evaluates the DFT on the points

$$\omega_k = k\Delta\omega, \qquad k = 0, 1, \ldots, N-1 \tag{4.1}$$

where $\Delta\omega$ is an arbitrary frequency spacing and N is the length of the input signal. For $\Delta\omega = 2\pi/N$ the FRFT evaluates the standard DFT. The frequencies, at which the FRFT evaluates the DFT, are illustrated in Fig. 2.

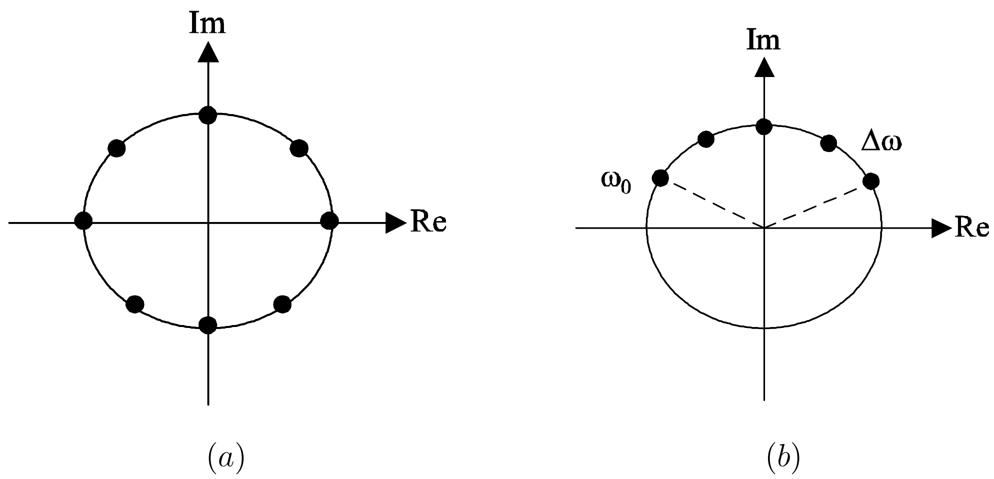


Figure 2: Frequency samples of the FFT and the Fractional FFT on the unit circle using N samples and a spacing $\Delta\omega$. (a) The FFT samples the Fourier transform over $[0, 2\pi]$, where $\Delta\omega = \frac{2\pi}{N}$. (b) The FRFT samples the Fourier transform using an arbitrary spacing $\Delta\omega$ and initial phase ω_0.

We denote by F_α the Centralized Fractional FFT (CFRFT), which computes the FRFT around the DC component. Thus, the CFRFT F_α "compresses" the FFT of the input signal around the DC component, where α is the compression ratio. Figure 2b is an example of CFRFT with $\alpha = 0.333$.

4.2 Computing the pseudo-polar FFT

We decompose the pseudo-polar grid into two sub-grids, denoted *Z* and *N*, as shown in Fig. 3. We denote by PP_Z and PP_N the values of the DFT evaluated on the sub-grids *Z* and *N*, respectively. We next describe the algorithm that computes PP_Z. The algorithm for PP_N is easily obtained by switching the roles of the X and Y axes. A thorough description of the algorithm is given in [20].

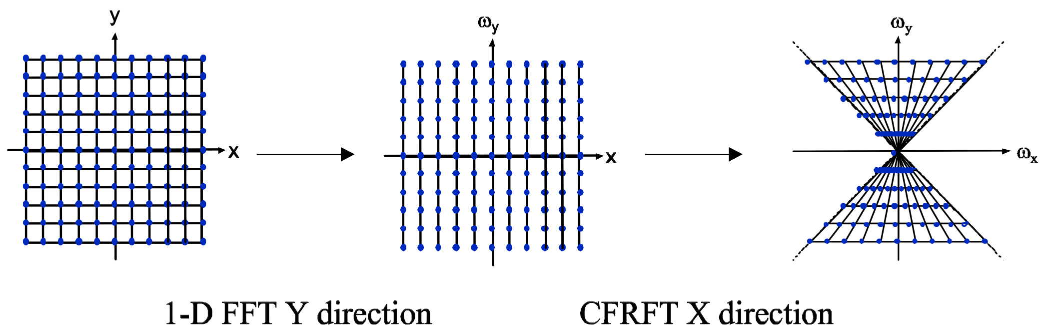


Figure 3: The construction of the pseudo-polar grid from the *Z* and *N* sub-grids.

4.2.1 Computing PP_Z

The computation of PP_Z is based on the separability of the 2D DFT. PP_Z is computed by applying the 1D FFT in the Y direction and then applying the CFRFT in the X direction. A varying α factor for the CFRFT is chosen such that the Z sub-grid geometry is obtained.

Algorithm Flow

1. Let I be the input image of size (n_0, m_0). I is zero padded to size $(n, n) = (2^k, 2^k)$ $k \in \mathbb{Z}$, where k satisfies $2^k \geq 2 \cdot Max\,(n_0, m_0)$.

2. Apply 1-D FFTs in the Y direction $I_{\widehat{x}} = FFT_X\,(I)$. Cyclically shift the result, such that the DC component is in the center of $I_{\widehat{x}}$. The shifting operation is similar to the "fftshift" command in Matlab™.

3. Apply the CFRFT operator F_α to the rows of $I_{\widehat{x}}$

$$PP_Z^i = F_{\alpha_i}\left(I_{\widehat{x}}^i\right) \tag{4.2}$$

where PP_Z^i is the ith row of PP_Z, F_{α_i} is the CFRFT operator with a compression ratio of α_i, $I_{\widehat{x}}^i$ is the ith row of $I_{\widehat{x}}$, and

$$\alpha_i = \begin{cases} \frac{n-2i}{n} & i \leq \frac{n}{2} \\ 0 & i = \frac{n}{2} \\ -\frac{n-2i}{n} & i > \frac{n}{2}. \end{cases} \tag{4.3}$$

To evaluate PP_N we transpose the input image I and reapply the above algorithm.

4.3 Geometric interpretation and properties

PP_Z and PP_N are matrices of size $n \times n$, whose elements are the values of the DFT on the PP grid. By examining Fig. 3 we see that each column in the matrices PP_Z and PP_N corresponds to a ray on the PP grid with a fixed angle θ. Similarly each row corresponds to some radius r. The polar and PP grids differ due the non-uniformity of the angular and radial spacings of the PP grid shown in Fig. 5. For the polar grid we have

$$\Delta\theta_{Polar}\,(i) = \frac{2\pi}{n}, \qquad \Delta r_{Polar}\,(j) = \Delta r_0$$

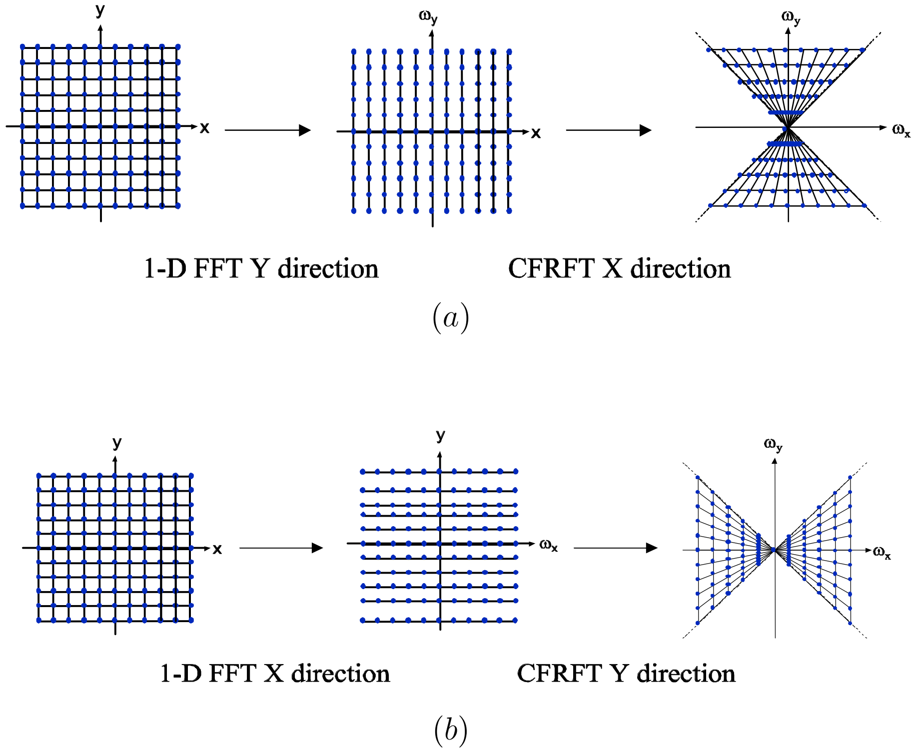


Figure 4: Computation of the pseudo-polar sub-grids. (a) The computation of the Z sub-grid. The input image is 1D FFT transformed in the Y direction. Then, the fractional FFT is applied on the X direction. (b) The computation of the N sub-grid. The input image is 1D FFT transformed in the X direction. Then, the fractional FFT is applied on the Y direction.

while for the PP grid, $\Delta\theta_{PP}(i)$ and $\Delta r_{PP}(j)$ vary smoothly as a function of i. Specifically, for the PP grid we have

$$\theta_{PP}(i) = \arctan\left(\frac{2i}{n}\right), \quad i = 0, \ldots, \frac{n}{2} \tag{4.4}$$

where n is the side of a $n \times n$ image and i is the index of the ray. Therefore,

$$\Delta\theta_{PP}(i) \triangleq \theta_{PP}(i+1) - \theta_{PP}(i) \tag{4.5}$$

$$\Delta r_{PP}(i) = \frac{\sqrt{\left(\frac{n}{2}\right)^2 + i^2}}{\left(\frac{n}{2}\right)} = \sqrt{1 + 4\left(\frac{i}{n}\right)^2}, \qquad i = 0, \ldots, \frac{n}{2}. \tag{4.6}$$

5 The angular difference function

The angular difference function (ADF) measures the difference between two images in the angular direction. This function was first presented in [23] for estimating large rotations between images.

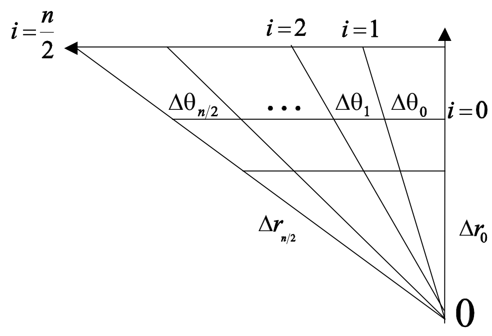


Figure 5: The geometrical properties of the PP grid. The angular and radial spacings $\Delta\theta_{PP}(i)$ and $\Delta r_{PP}(i)$, respectively, vary smoothly as a function of i.

In section 5.1 we present a method for 1D shift estimation using difference functions. In section 5.2 we present the application of the ADF to the frequency domain of 2D images. We conclude the presentation of the ADF in section 5.3, by presenting a fast and accurate algorithm for its computation.

5.1 Translation estimation using difference functions

We begin the derivation of the ADF with a 1D example. Difference functions (DF) enable us to derive a naive algorithm for 1D shift estimation. Let $f_1(x)$ and $f_2(x)$, $x \in [0, N]$, be two shifted versions of the same function. Specifically, $f_1(x) = f_2(x + \Delta x)$ (See Fig. 6a). We denote by $g_2(x)$ the flipped and shifted version of $f_2(x)$

$$g_2(x) = f_2(-x + N) \tag{5.1}$$

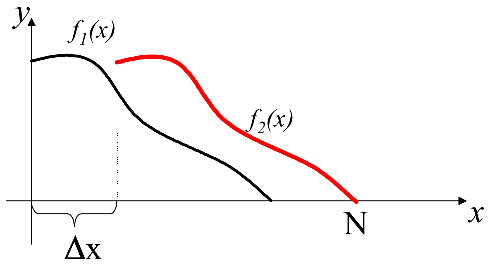
(see Fig.6b). We define the difference function (DF) Δf by

$$\begin{aligned} \Delta f(x) &= f_1(x) - g_2(x) \\ &= f_1(x) - f_2(-x + N) \\ &= f_2(x + \Delta x) - f_2(-x + N) \end{aligned} \tag{5.2}$$

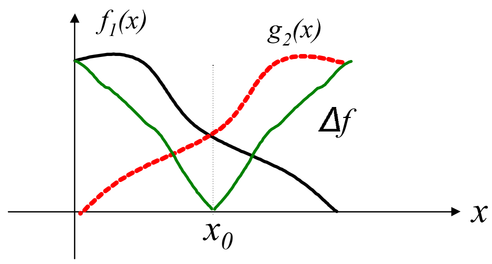
and consider its zeros $\Delta f(x) = 0$. One of its zeros necessarily satisfies

$$\begin{aligned} x_0 + \Delta x &= -x_0 + N \\ \Delta x &= \frac{N}{2} - x_0 \end{aligned} \tag{5.3}$$

which means that we can estimate the relative translation from the location of the zero of Δf. Equation (5.3) holds for arbitrarily sampled functions $f_1(x)$ and $f_2(x)$. In such a case, instead of searching for the zero of Δf, we search for the minimum of $|\Delta f|$. In general, Eq. (5.3) does not have a unique solution.



(a)



(b)

Figure 6: Translation estimation using the angular difference function. Given translated input signals $f_1(x)$ and $f_2(x)$ (a), the translation is estimated (b) by flipping $f_2(x)$ and computing the difference function Δf, whose zero cooresponds to twice the shift.

5.2 The difference function in the Fourier domain

In this section we derive the difference function for 2D images. Given two images I_1 and I_2, we denote by $M_1(r,\theta)$ and $M_2(r,\theta)$ the magnitudes of the Fourier transforms of I_1 and I_2, respectively. If I_2 is a rotated and translated version of I_1, i.e.,

$$I_2(x,y) = I_1(x\cos\Delta\theta + y\sin\Delta\theta + \Delta x, -x\sin\Delta\theta + y\cos\Delta\theta + \Delta y) \tag{5.4}$$

then

$$M_1(r,\theta) = M_2(r,\theta+\Delta\theta). \tag{5.5}$$

We define the difference function of $M_1(r,\theta)$ and $M_2(r,-\theta)$ in the angular direction by

$$\Delta M(\theta) = \int_0^\infty |M_1(r,\theta) - M_2(r,-\theta)|\,dr, \qquad \theta\in[0,\pi]. \tag{5.6}$$

The value of $\Delta M(\theta_0)$ is zero if

$$\theta_0 + \Delta\theta = -\theta_0 \qquad \text{or} \qquad \theta_0 + \Delta\theta = -\theta_0 + \pi \tag{5.7}$$

where the second zero is due to the conjugate symmetry of M_1 and M_2. Thus, we get that the two zeros of $\Delta M(\theta)$, obtained at θ_0^1 and θ_0^2, are related to the relative rotation $\Delta\theta$ by

$$\theta_0^{(1)} = -\frac{\Delta\theta}{2}, \qquad \theta_0^{(2)} = -\frac{\Delta\theta}{2} + \frac{\pi}{2}. \tag{5.8}$$

We see from Eq. (5.8) that the zeros θ_0^1 and θ_0^2 are $\pi/2$ radians apart. This property is true in general. For each zero θ_0 of ΔM, $\theta_0 + \frac{\pi}{2}$ is also a zero. Therefore, we define the angular difference function (ADF) by

$$ADF(\theta) = \Delta M(\theta) + \Delta M\left(\theta + \frac{\pi}{2}\right) \qquad \theta \in \left[0, \frac{\pi}{2}\right]. \tag{5.9}$$

The zero θ_0 of $ADF(\theta)$ is related to the rotation angle $\Delta\theta$ by

$$\theta_0^{(1)} = -\frac{\Delta\theta}{2}. \tag{5.10}$$

Note that since the we compute the ADF using the magnitude of the Fourier transform, it is invariant to any translations of the input images.

5.3 Computing the ADF for discrete images

An important property of $\Delta M(\theta)$ is that it can be discretized using very general sampling grids. The only requirement from the sampling grid is that if θ is a sampling point, then, $\theta + \frac{\pi}{2}$ is also a sampling point. Therefore, to compute the ADF we do not need a true polar representation of the Fourier transforms of I_1 and I_2.

The reversal of the angular axis, indicated by Eq. (5.6), is accurately implemented by flipping the input image either along the x or the y axes. Mathematically,

$$\widetilde{I}(x,y) = fliplr(I(x,y)) \quad \Leftrightarrow \quad \widetilde{I}(r,\theta) = I(r,-\theta). \tag{5.11}$$

The PPFT, presented in Section 4, is used to derive a fast and accurate algorithm for the computation of the ADF. The PPFT evaluates the DFT of an image over a non-uniform polar grid. For each angle θ in the PP grid, the grid also contains the angle $\theta + \frac{\pi}{2}$. Thus, we use the PPFT algorithm to compute the ADF as follows: Given input images I_1 and I_2, defined on a Cartesian grid,

1. Flip I_1 in the left→right direction.

2. Compute M_1^d and M_2^d, where M_j^d is the magnitude of the PPFT of I_j, $j = 1, 2$.

3. Evaluate Eq. (5.6) using numerical integration

$$\Delta M^{d}(\theta_i) = \sum_{0 \leq r_j \leq \pi} \left| M_1^d(r_j, \theta_i) - M_2^d(r_j, -\theta_i) \right| \Delta r_i, \qquad \theta_i \in [0, \pi]. \tag{5.12}$$

   Note that the integration is computed over rays of the same length, where Δr_i is the radial sampling interval. See Fig. 7.

4. Compute the ADF by

$$ADF(\theta_i) = \Delta M^{d}(\theta_i) + \Delta M^{d}(\theta_{i+K}) \tag{5.13}$$

   where K is the size of the PP grid.

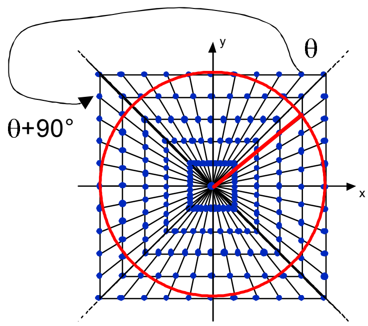


Figure 7: Computing the ADF using the PPFT. The ADF is computed by evaluating $\Delta M^{d}(\theta_i)$ over rays in the PP grid. The integration is computed only inside the unit circle. Given a ray at angle θ, the ray at angle $\theta + \frac{\pi}{2}$ is also in the PP grid. This enables computing the function $ADF(\theta) = \Delta M^{d}(\theta) + \Delta M^{d}\left(\theta + \frac{\pi}{2}\right)$.

6 Symmetry detection algorithm

The symmetry detection algorithm consists of three stages. First, in section 6.1, the algorithm determines N, the number of ADF minima . Then, in section 6.2, it uses N to find the symmetry axes. Finally, for non-centered symmetries, it locates the center of the symmetry as described in section 6.3.

6.1 Computing the number of minima of the ADF

For a given input image $I(x, y)$, we detect reflectional symmetry by setting

$$\begin{aligned} I_1(x, y) &= I(x, y) \\ I_2(x, y) &= fliplr(I(x, y)) \end{aligned} \tag{6.1}$$

and computing the ADF of I_1 and I_2 according to section 5.3. Similarly, to detect rotational symmetry we set

$$\begin{aligned} I_1(x, y) &= I(x, y) \\ I_2(x, y) &= I(x, y). \end{aligned} \tag{6.2}$$

and again, compute the ADF of I_1 and I_2. According to Eq. (5.5), translations of the input image do not change the ADF, and therefore, the number of minima of the ADF is the same for centered and non-centered images. Figure 8 illustrates the ADF of a symmetric image. We can clearly observe that the number of minima corresponds to the degree of symmetry in the input image.

We will robustly estimate the number of minima in the ADF by using its spectrum, denoted by S_{ADF}. If the ADF has N minima, then the spectrum S_{ADF} has a maximum at ω_N. The number of minima N is given by

$$N = \arg\max_i S_{ADF}(\omega_i) \tag{6.3}$$

In the example shown in Fig. 8 there are 3 symmetry axes. Thus, the maxima of the spectrum S_{ADF} will be detected at ω_3, which is the third coefficient following the DC coefficient. The index in which the spectrum S_{ADF} achieves its maximum is invariant to the size of the input image, as the maximum of S_{ADF} counts the number of maxima in the ADF and does not relate to the size of the input image.

Natural and synthetic objects usually exhibit low symmetry orders, e.g., $N < 15$, which makes ω_N a very low frequency. Thus, the ADF can be pre-processed by low-pass filtering. Figure 9 demonstrates the analysis of the *Pentagon* image, where the ADF is more noisy than the ADF of the synthetic image given in Fig. 8.

Since the ADF is defined over a non-uniform abscissa in the θ direction, we need to resample it on a uniform θ axis before using any lowpass filtering. Then, we can compute the S_{ADF} using

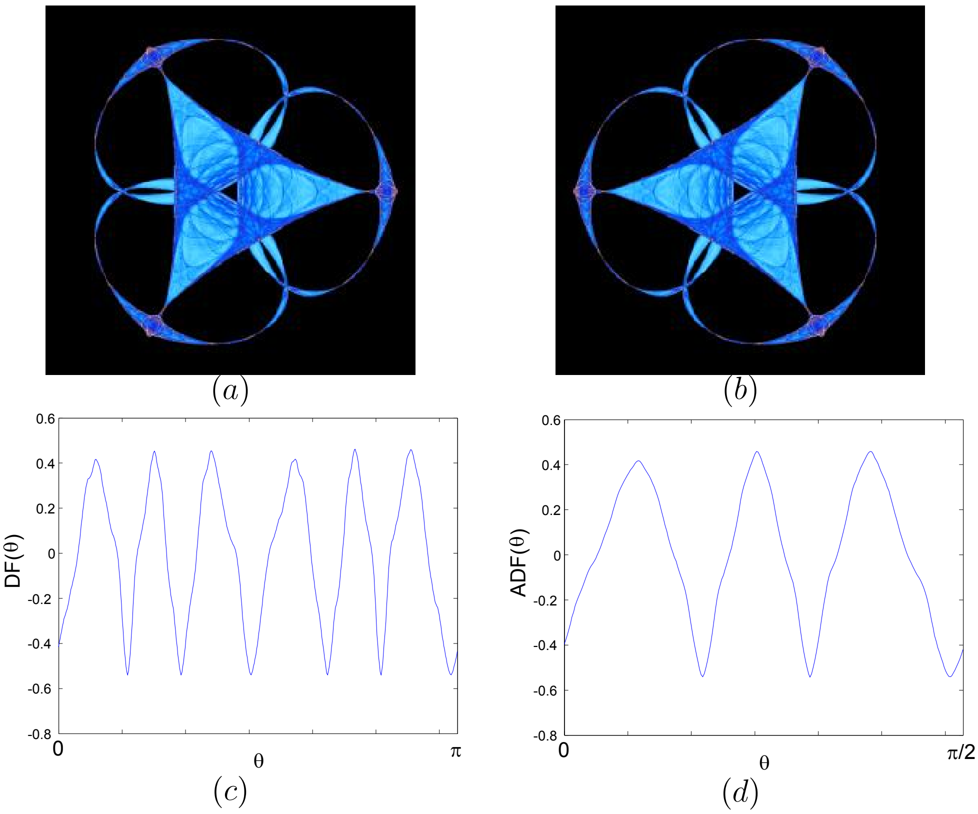


Figure 8: The ADF of a symmetric image. The input image (a) is flipped left→right to create (b). ΔM in (c) is the difference function of (a) and (b). The ADF in (d) is computed by averaging ΔM with an offset of $\frac{\pi}{2}$.

nonparametric spectrum estimation [24, 25, 26]. Figure 10b shows the spectrum S_{ADF} of the *Pentagon* image computed using the MUSIC algorithm [24]. Computing the spectrum S_{ADF} simply by using the magnitude of the FFT results in detecting a wrong number of minima (Fig. 10b).

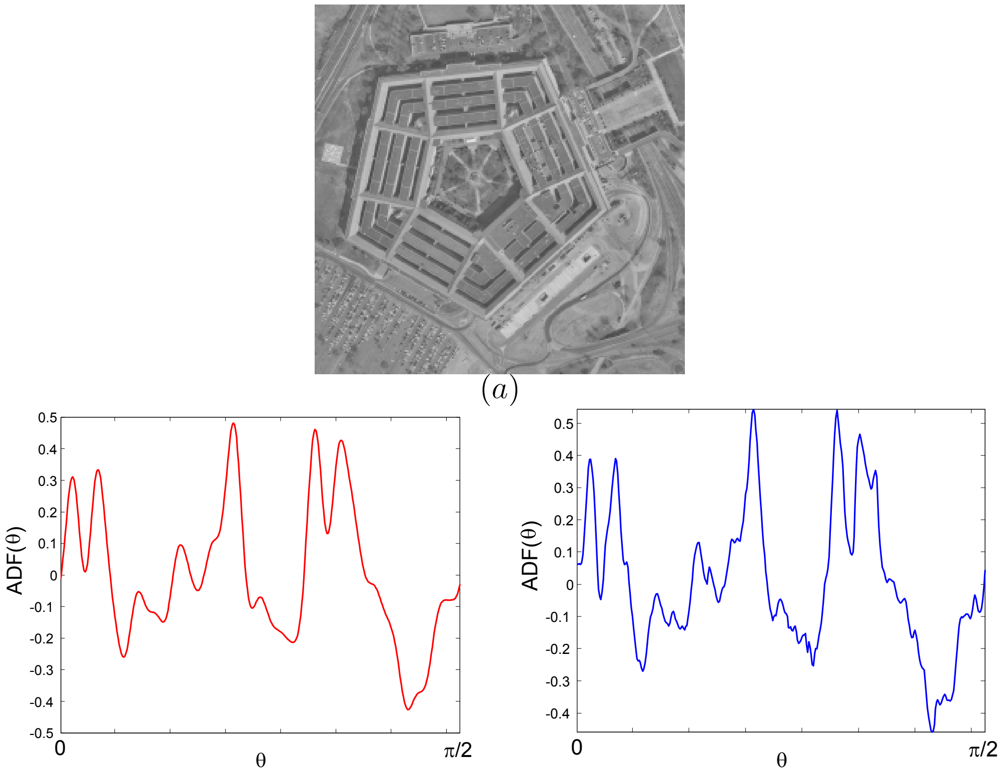


Figure 9: Lowpass prefiltering the ADF. In real images such as the *Pentagon* (a) the ADF might be noisy (b). It can be cleaned by lowpass filtering (c) .

6.2 Computing the symmetry axes

The directions of the symmetry axes are given by

$$\theta_s(i, N_s) = \frac{2\pi}{N_s} i + \theta_0, \qquad i = 0, .., N_s - 1 \tag{6.4}$$

(see Eq. (3.5)), where θ_0 is the angle of any of the symmetry axes and N_s is the number of symmetry axes. Note that N_s is not necessarily equal to N, which we computed in section 6.1. In section 6.2.1 we describe how to compute θ_0, and in section 6.2.2 we describe the computation of N_s.

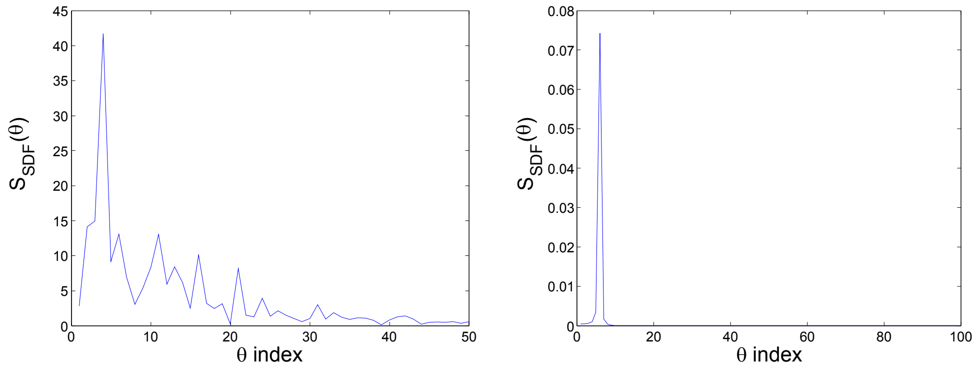


Figure 10: S_{ADF} estimation. (a) Using the FFT to estimate S_{ADF} results in a noisy estimate. (b) The MUSIC non-parametric spectrum estimation algorithm give a more accurate estimation.

6.2.1 Computing θ_0

For an image with N symmetry axes, the ADF, given in Eq. (5.13), has N minima. Moreover, the ADF is periodic with period $\frac{L_{ADF}}{N}$, where L_{ADF} is the length of the ADF and N is given in section 6.1. Therefore, a single period of the ADF,denoted ADF_p,is computed by

$$ADF_p(i) = \sum_{j=1}^{N} ADF(i + jT), \qquad i = 0, \ldots, T - 1. \tag{6.5}$$

This summation creates a dominant minimum at a point $\Delta\theta$. See for example Fig. 11 for the ADF_P of the *Pentagon* image. The angle θ_0 is given by

$$\theta_0 = -\frac{\Delta\theta}{2} \tag{6.6}$$

where

$$\Delta\theta = \arg\min_{\theta} ADF_p(\theta_i). \tag{6.7}$$

The angle $\Delta\theta$, computed by Eq. (6.7), is equal to the relative rotation angle given in Eq. (5.5). Due to conjugate symmetry [23, 27], relative rotations by $\Delta\theta$ and $\Delta\theta + \pi$ result in the same ADF. We therefore need to resolve the actual value of $\Delta\theta$ in order to compute θ_0. This is done by rotating $I(x, y)$ by $\Delta\theta$ and $\Delta\theta + \pi$ and choosing the angle that gives largest correlation with $fliplr(I(x, y))$.

6.2.2 Computing the number of symmetry axes

If the number of minima of the ADF is odd, then the number of symmetry axes N_s is equal to the number of minima N, given by Eq. (6.3). If the number of minima N is even, then either $N_s = N$

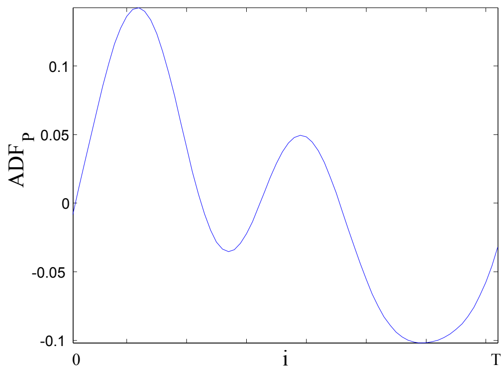


Figure 11: Computing ADF_P. Given the number of minima N, we compute ADF_P from ADF. The global minimum of ADF_P corresponds to the angle of one of the symmetry axes.

or $N_s = 2N$ [3]. As can be seen from Eq. (5.8), we cannot distinguish between symmetry axes at angles θ and $\theta + \pi/2$. Both angles result in a minimum in the ADF at angle θ. Therefore, the angles

$$\theta_s(i, N) = \frac{2\pi}{N} i + \theta_0, \qquad i = 0, .., N-1 \tag{6.8}$$

are guaranteed to be angles of symmetry axes . Remains to determine whether $\theta_s(2i+1, 2N)$ are also symmetry axes. This is done by comparing the registration errors related to $\theta_s(1, k)$ and $\theta_s(1, 2k)$.

6.3 Detecting the center of symmetry

For non-centered symmetries, we detect the center of symmetry by computing two symmetry axes and finding their intersection point (x_0, y_0). This procedure is illustrated for the *pentagon* image in Fig. 12.

The first symmetry axis l_1 is determined as follows. We rotate the image I by θ_0, given by Eq. (6.6), around the center of the image. We denote the rotated image I_1. The symmetry axis of I_1 is now parallel to the y axis. Hence, we can compute its location by flipping I_1 along the y axis and estimating the 1D translation Δx, along the x axis, between I_1 and its flipped version. The location of the vertical axis $\widetilde{l_1}$ is then given by $\frac{\Delta x}{2}$. We compute the axis l_1 by rotating $\widetilde{l_1}$ around the center of I by $-\theta_0$.

Similarly, we find l_2 by rotating I by $\theta_0 + \frac{2\pi}{N_s}$ and repeating the above procedure. The symmetry center (x_0, y_0) is then given by the intersection point of l_1 and l_2 (Fig. 12a). Using the symmetry center we determine all other symmetry axes (Fig. 12b).

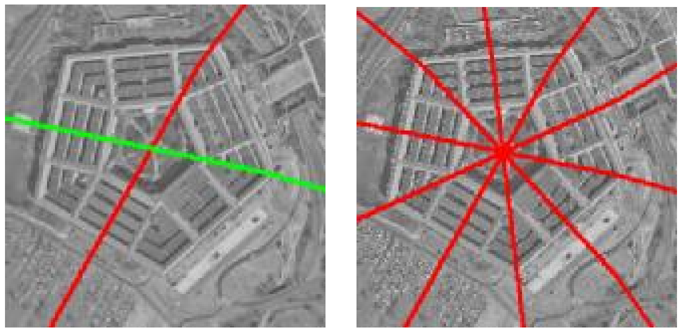


Figure 12: Symmetry center detection. The symmetry center is detected by computing the parameters of two symmetry axes (a) and finding the symmetry center. The rest of the symmetry axes can then be computed (b) .

7 Experimental results

The proposed algorithm was extensively tested using synthetic and real images. For each image we present its ADF, its lowpass filtered ADF, and its spectrum S_{ADF}. For reflectional symmetries, the detected symmetry axes are overlayed on the input image. For all images, the spectrum S_{ADF} was computed using a four dimensional MUSIC algorithm without zero padding. The prefiltering was performed by multiplying the FFT of the ADF with the window shown in Fig. 13, whose pass range is of length 30.

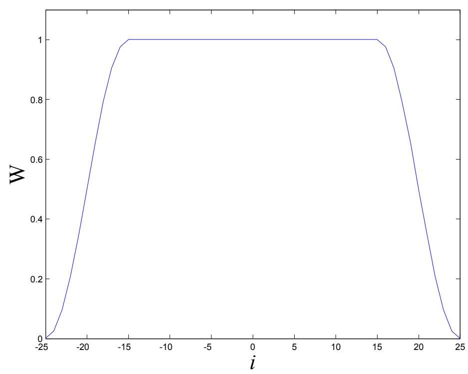
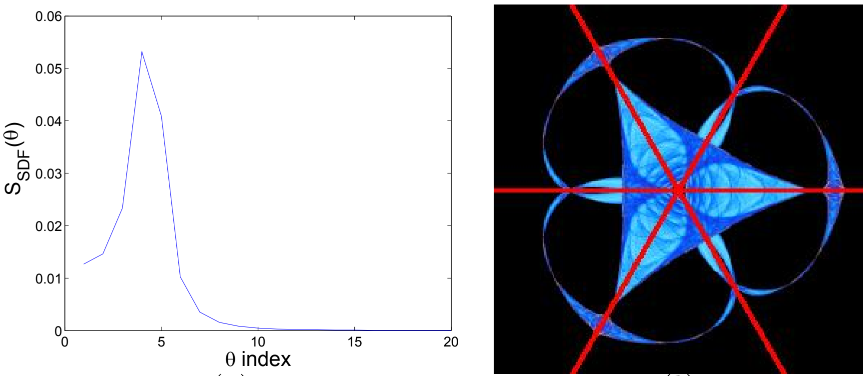


Figure 13: Fourier domain windowing. The window is used to window the ADF. The bandpass range is $[-15, 15]$.

For the synthetic images, given in Figs. 14, 15, 16, and 17, we can clearly see the periodic nature of the ADF. Thus, for synthetic images there is no need to lowpass filter the ADF. The number of symmetry axes and their exact location are clearly detected.

In Figs. 9, 18 and 19 we apply the algorithm to real images. Figure 19 shows a non-symmetric image used to test the performance of the algorithm for non-symmetric images. We can see that for Figs. 9 and 18 the algorithm detected the correct number of symmetry axes although they contain significant non-symmetric parts.



(a) (b)

Figure 14: D_3 symmetry detection results. (a) The S_{ADF} clearly show a peak at $k = 3$. (b) The detected symmetry axes. The ADF and filtered ADF are given in Figs. 8c and 8d, respectively.

8 Conclusions

We presented a 2D symmetry detection algorithm, which detects both rotational and reflectional symmetries. The algorithm is based on the ADF, which measures the difference between the Fourier transforms of two images in the angular direction. The ADF can be computed accurately and rapidly using the pseudo-polar Fourier transform. When applied to two copies of the same image, the zeros of the ADF are shown to correspond to the symmetry axes of the image. We demonstrated the applicability of the algorithm to both synthetic and real-life images, including noisy and non-centered images.

9 Acknowledgements

The algorithm was tested using patterns kindly provided by Prof. Michael Field. These patterns were created by students in the course “Patterns, Designs, and Symmetries” taught by Prof. Field in the Department of Art at the University of Houston, Texas.

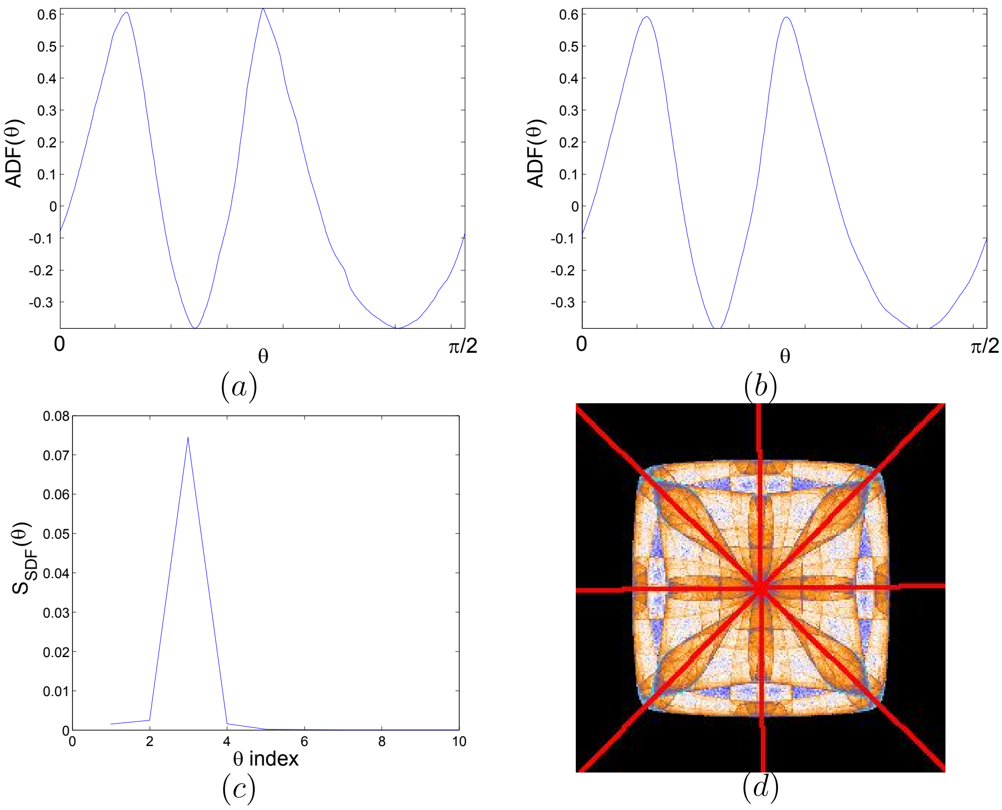


Figure 15: D_4 symmetry detection results. (a) The ADF before filteration. (b) The ADF after lowpass filtering. There is no significant improvement as the input image is synthetic. (c) The S_{ADF}. (d) Detected symmetry axes.

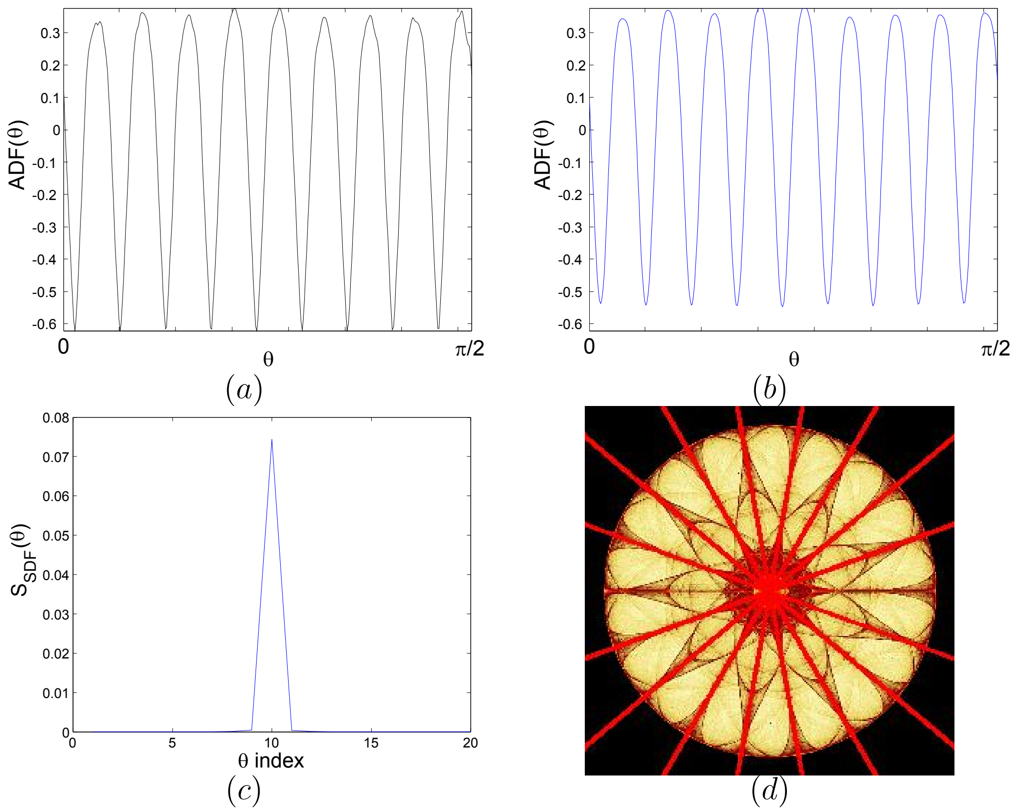


Figure 16: D_9 symmetry detection results. (a) The ADF before filteration. (b) The ADF after lowpass filtering. There is no significant improvement as the input image is synthetic. (c) The S_{ADF}. (d) Detected symmetry axes.

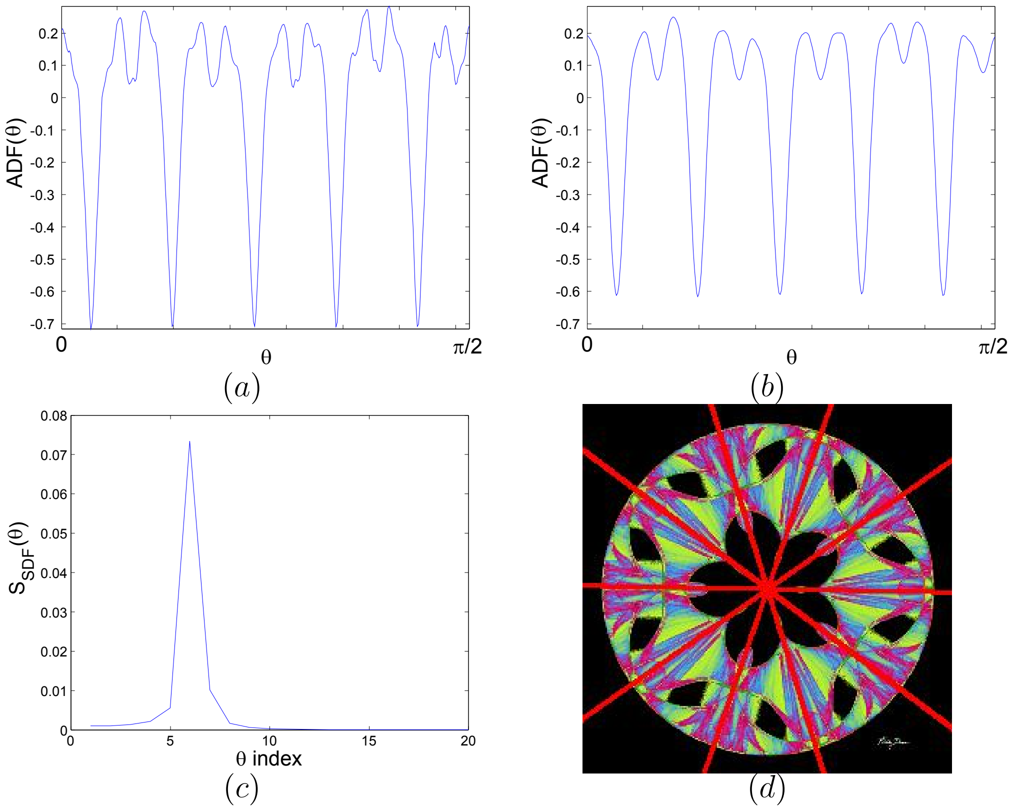


Figure 17: C_5 symmetry detection results. (a) The ADF before filteration. (b) The ADF after lowpass filtering. There is no significant improvement as the input image is synthetic. (c) The S_{ADF}. (d) Detected symmetry axes.

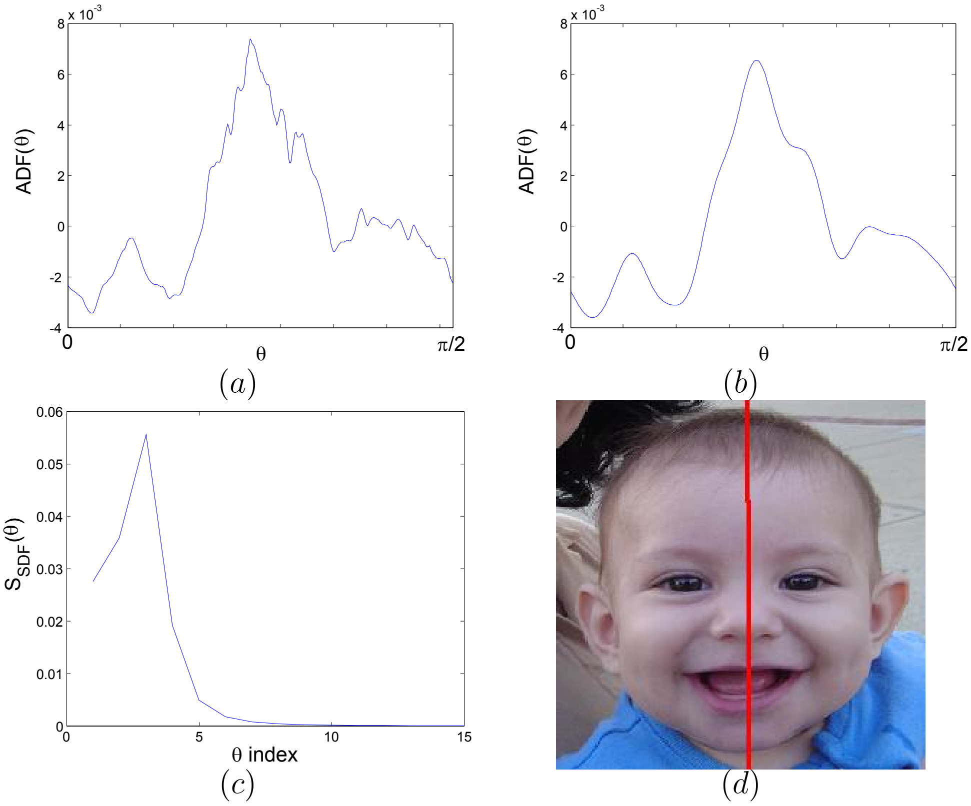


Figure 18: D_1 symmetry axes detection for *Baby* image. (a) The ADF of a real image is noisy due to non-symmetric parts. (b) Filterd ADF. For real images the filtration results in a significant improvement. (c) The S_{ADF}. (d) Detected symmetry axes.

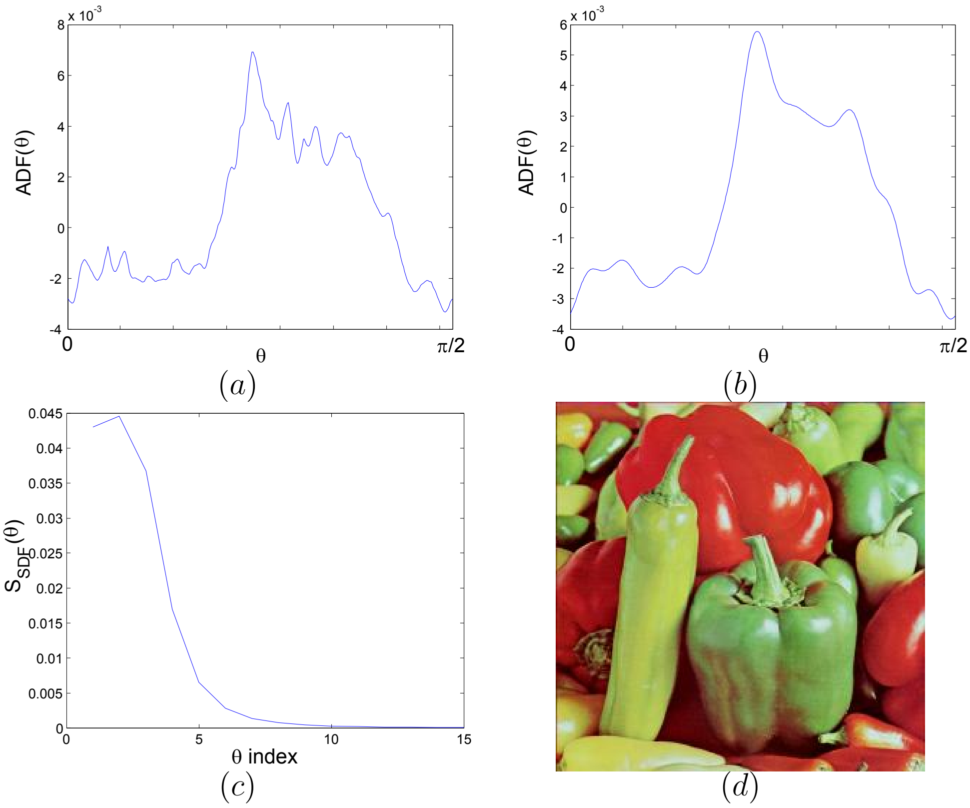


Figure 19: Symmetry analysis of a non-symmteric image. (a) The ADF of a non-symmetric image is noisy and non periodic, even after filtering (b). (c) There is no significant peak of the S_{ADF}. The non symmetry can be verified by computing the alignment error related to the maxima of the S_{ADF}. (d) Non symmetric input image.

References

[1] Luca Lucchese. A frequency domain algorithm for dectection and classification of cyclic and dihedral symmetries in two-dimensional patterns. In *of Int'l Conference on Image Processing (ICIP)*, volume II, pages 793–796, Rochester, NY, September 2002.

[2] W. Miller. *Symmetry Groups and their Applications*. Academic Press, London, 1972.

[3] H. Weyl. *Symmetry*. Princeton University Press, 1952.

[4] Daniel Reisfeld, Haim Wolfson, and Yehezkel Yeshurun. Context free attentional operators: the generalized symmetry transform. *International Journal of Computer Vision*, pages 119–130, 1995.

[5] Nahum Kiryati and Yossi Gofman. Detecting symmetry in grey level images: The global optimization approach. *International Journal of Computer Vision*, 29(1):29–45, August 1998.

[6] Hagit Zabrodsky, Shmuel Peleg, and David Avnir. Symmetry as a continuous feature. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1154–1166, 1995.

[7] Whoi-Yul Kim and Young-Sung Kim. Robust rotation angle estimator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):768–773, August 1999.

[8] Michael M. Kazhdan, Bernard Chazelle, David P. Dobkin, Adam Finkelstein, and Thomas A. Funkhouser. A reflective symmetry descriptor. In *ECCV (2)*, pages 642–656, 2002.

[9] Changming Sun and Jamie Sherrah. 3d symmetry detection using the extended gaussian image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):164–168, February 1997.

[10] Dinggang Shen, Horace H. S. Ip, and Eam Khwang Teoh. Robust detection of skewed symmetries by combining local and semi-local affine invariants. *Pattern Recognition*, 34(7):1417–1428, 2001.

[11] L. Lucchese and G.M. Cortelazzo. A noise-robust frequency domain technique for estimating planar roto-translations. *IEEE Transactions on Signal Processing*, 48(6):1769–1786, June 2000.

[12] B. Johansson, H. Knutsson, and G. Granlund. Detecting rotational symmetries using normalized convolution. *In Proceedings of the 15th International Conference on Pattern Recognition*, 3:500–504, September 2000.

[13] Si-Duo Chen. Extraction of local mirror-symmetric feature by odd-even decomposition. *Proceedings International Conference on Image Processing*, 3:756 – 759, 2001.

[14] F. Labonte, Y. Shapira, and P. Cohen. A perceptually plausible model for global symmetry detection. *Proceedings Fourth International Conference on Computer Vision*, pages 258 – 263, May 1993.

[15] T-L Liu, D. Geiger, and A.L. Yuille. Segmenting by seeking the symmetry axis. *14th International Conference on Pattern Recognition. Australia.*, August 1998.

[16] Gareth Loy and Alexander Zelinsky. Fast radial symmetry for detecting points of interest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(08):959–973, 2003.

[17] Dong-Su Kim and Sung-Il Chien. Automatic car license plate extraction using modified generalized symmetry transform and image warping. *IEEE Proceedings International Symposium on Industrial Electronics*, 3(3):2022 – 2027, June 2001.

[18] James B. Hayfron-Acquah, Mark S. Nixon, and John N. Carter. Human identification by spatio-temporal symmetry. *In Proceedings International Conference on Pattern Recognition*, pages 632–635, 2002.

[19] Kuo-Chin Fan, Wei-Hsien Wu, and Meng-Pang Chung. A symmetry-based coarse classification method for chinese characters. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 32(4):522 – 528, November 2002.

[20] R.R Coifman A. Averbuch, D.L. Donoho, M. Israeli, and Y. Shkolnisky. Fast slant stack: A notion of Radon transform for data in cartesian grid which is rapidly computable, algebraically exact, geometrically faithful and invertible. *SIAM Scientific Computing*, To appear.

[21] David H. Bailey and Paul N. Swarztrauber. The fractional fourier transform and applications. *SIAM Review*, 33(3):389–404, September 1991.

[22] L.R. Rabiner, R.W. Schafer, and C.M. Rader. The chirp z-transform algorithm. *IEEE Transcations on Audio ElectroScoustics*, AU(17):86–92, June 1969.

[23] Yosi Keller, Yoel Shkolnisky, and Amir Averbuch. Alebraically accurate rotation estimation. Technical Report 2908, Tel-Aviv University, 2003.

[24] E. J. Hannan B. G. Quinn. *The Estimation and Tracking of Frequency*. Cambridge University Press, 2001.

[25] A. Oppenheim and R. Schafer. *Disrete-Time Signal Processing.* Prentice Hall, 89.

[26] Boaz Porat. *A Course in Digital Signal Processing*. John Wiley Pub., 1997.

[27] S. Reddy and B. N. Chatterji. An fft-based technique for translation, rotation, and scale-invariant image registration. *IEEE Trans. on Image Processing*, 3(8):1266–1270, August 1996.