

---

# PyEFD Documentation

*Release 1.0*

**Henrik Blidh**

**Jul 27, 2019**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>
2.1	General usage examples . . . . .	5
2.2	OpenCV example . . . . .	5
2.3	Using EFD as features . . . . .	6
<b>3</b>	<b>Testing</b>	<b>7</b>
<b>4</b>	<b>References</b>	<b>9</b>
<b>5</b>	<b>API</b>	<b>11</b>
5.1	References . . . . .	11
<b>6</b>	<b>Indices and tables</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>
	<b>Index</b>	<b>17</b>



An Python/NumPy implementation of a method for approximating a contour with a Fourier series, as described in<sup>1</sup>.

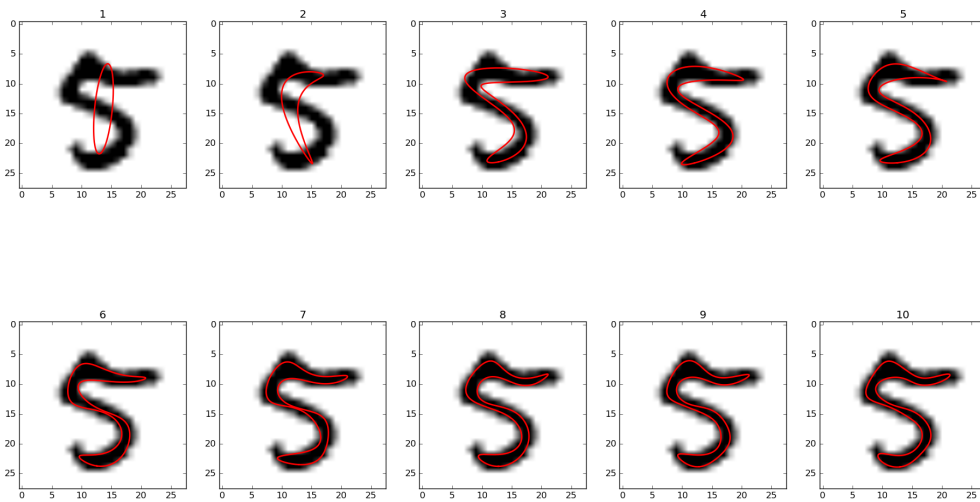


Fig. 1: EFD representations of an MNIST<sup>2</sup> digit. Shows progressive improvement of approximation by order of Fourier series.

<sup>1</sup> Frank P Kuhl, Charles R Giardina, Elliptic Fourier features of a closed contour, Computer Graphics and Image Processing, Volume 18, Issue 3, 1982, Pages 236-258, ISSN 0146-664X, [http://dx.doi.org/10.1016/0146-664X\(82\)90034-X](http://dx.doi.org/10.1016/0146-664X(82)90034-X).

<sup>2</sup> LeCun et al. (1999): The MNIST Dataset Of Handwritten Digits



# CHAPTER 1

---

## Installation

---

```
$ pip install pyefd
```





Given a closed contour of a shape, generated by e.g. `scikit-image` or `OpenCV`, this package can fit a `Fourier series` approximating the shape of the contour.

## 2.1 General usage examples

This section describes the general usage patterns of `pyefd`.

```
from pyefd import elliptic_fourier_descriptors
coeffs = elliptic_fourier_descriptors(contour, order=10)
```

The coefficients returned are the  $a_n$ ,  $b_n$ ,  $c_n$  and  $d_n$  of the following Fourier series representation of the shape.

The coefficients returned are by default normalized so that they are rotation and size-invariant. This can be overridden by calling:

```
from pyefd import elliptic_fourier_descriptors
coeffs = elliptic_fourier_descriptors(contour, order=10, normalize=False)
```

Normalization can also be done afterwards:

```
from pyefd import normalize_efd
coeffs = normalize_efd(coeffs)
```

## 2.2 OpenCV example

If you are using `OpenCV` to generate contours, this example shows how to connect it to `pyefd`.

```
import cv2
import numpy
from pyefd import elliptic_fourier_descriptors
```

(continues on next page)

(continued from previous page)

```
# Find the contours of a binary image using OpenCV.
contours, hierarchy = cv2.findContours(
    im, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Iterate through all contours found and store each contour's
# elliptical Fourier descriptor's coefficients.
coeffs = []
for cnt in contours:
    # Find the coefficients of all contours
    coeffs.append(elliptic_fourier_descriptors(
        numpy.squeeze(cnt), order=10))
```

## 2.3 Using EFD as features

To use EFD as features, one can write a small wrapper function:

```
def efd_feature(contour):
    coeffs = elliptic_fourier_descriptors(
        contour, order=10, normalize=True)
    return coeffs.flatten()[3:]
```

If the coefficients are normalized, then `coeffs[0, 0] = 1.0`, `coeffs[0, 1] = 0.0` and `coeffs[0, 2] = 0.0`, so they can be disregarded when using the elliptic Fourier descriptors as features.

See<sup>1</sup> for more technical details.

## CHAPTER 3

---

### Testing

---

Run tests with:

```
$ python setup.py test
```

or with `Pytest`:

```
$ py.test tests.py
```

The tests includes a single image from the MNIST dataset of handwritten digits (<sup>2</sup>) as a contour to use for testing.



## CHAPTER 4

---

References

---



A Python implementation of the method described in<sup>3</sup> and<sup>4</sup> for calculating Fourier coefficients for characterizing closed contours.

## 5.1 References

Created by hblidh <[henrik.blidh@nedomkull.com](mailto:henrik.blidh@nedomkull.com)> on 2016-01-30.

`pyefd.calculate_dc_coefficients` (*contour*)

Calculate the  $A_0$  and  $C_0$  coefficients of the elliptic Fourier series.

**Parameters** `contour` (*numpy.ndarray*) – A contour array of size  $[M \times 2]$ .

**Returns** The  $A_0$  and  $C_0$  coefficients.

**Return type** tuple

`pyefd.elliptic_fourier_descriptors` (*contour*, *order=10*, *normalize=False*)

Calculate elliptical Fourier descriptors for a contour.

**Parameters**

- **contour** (*numpy.ndarray*) – A contour array of size  $[M \times 2]$ .
- **order** (*int*) – The order of Fourier coefficients to calculate.
- **normalize** (*bool*) – If the coefficients should be normalized; see references for details.

**Returns** A  $[order \times 4]$  array of Fourier coefficients.

**Return type** `numpy.ndarray`

---

<sup>3</sup> F. P. Kuhl and C. R. Giardina, “Elliptic Fourier Features of a Closed Contour,” *Computer Vision, Graphics and Image Processing*, Vol. 18, pp. 236-258, 1982.

<sup>4</sup> Oivind Due Trier, Anil K. Jain and Torfinn Taxt, “Feature Extraction Methods for Character Recognition - A Survey”, *Pattern Recognition* Vol. 29, No.4, pp. 641-662, 1996

`pyefd.normalize_efd` (*coeffs*, *size\_invariant=True*)

Normalizes an array of Fourier coefficients.

See<sup>3</sup> and<sup>4</sup> for details.

**Parameters**

- **coeffs** (*numpy.ndarray*) – A [n x 4] Fourier coefficient array.
- **size\_invariant** (*bool*) – If size invariance normalizing should be done as well. Default is `True`.

**Returns** The normalized [n x 4] Fourier coefficient array.

**Return type** `numpy.ndarray`

`pyefd.plot_efd` (*coeffs*, *locus=(0.0, 0.0)*, *image=None*, *contour=None*, *n=300*)

Plot a [2 x (N / 2)] grid of successive truncations of the series.

---

**Note:** Requires `matplotlib`!

---

**Parameters**

- **coeffs** (*numpy.ndarray*) – [N x 4] Fourier coefficient array.
- **tuple or numpy.ndarray locus** (*list,*) – The  $A_0$  and  $C_0$  elliptic locus in<sup>3</sup> and<sup>4</sup>.
- **n** (*int*) – Number of points to use for plotting of Fourier series.



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**p**

`pyefd`, 11



## C

`calculate_dc_coefficients()` (*in module pyefd*), 11

## E

`elliptic_fourier_descriptors()` (*in module pyefd*), 11

## N

`normalize_efd()` (*in module pyefd*), 11

## P

`plot_efd()` (*in module pyefd*), 12

`pyefd` (*module*), 11