Haar Wavelet

> 计64 翁家翌 2016011446

小波变换

- 母小波函数$\psi(t)$必须满足下列条件：
  - $\int_{-\infty}^{\infty}|\psi(t)|^2dt=1$，也即$\psi\in L^2(\mathbb{R})$并单位化
  - $\int_{-\infty}^{\infty}|\psi(t)|dt<\infty$，也即$\psi\in L^1(\mathbb{R})$
  - $\int_{-\infty}^{\infty}\psi(t)dt=0$
- 小波变换形式：

$$X(a,b)=\frac{1}{\sqrt{b}}\int_{-\infty}^{\infty}x(t)\Psi(\frac{t-a}{b})dt$$

  其中b是尺度，a是平移量

- 分为离散小波变换和连续小波变换
  - 连续：在所有可能的缩放和平移上进行操作
  - 离散：采用特定的缩放和平移值

哈尔小波变换

- 是一种离散小波变换
- 一维序列：
  - 64, 2, 3, 61, 60, 6, 7, 57
  - step 1, mean: 33, 32, 33, 32; diff: 31, -29, 27, -25
  - step 2, mean: 32.5, 32.5; diff: 0.5, 0.5
  - step 3, mean: 32.5; diff: 0
  - thus the final result of Haar: 32.5, 0, 0.5, 0.5, 31, -29, 27, -25
- 二维图像：
  - step 1：切割成8*8
  - step 2：对每个8*8，做一次一维行变换
  - step 3：对每个8*8，做一次一维列变换
  - step 4：将绝对值$\leq\delta$的数字以0填充

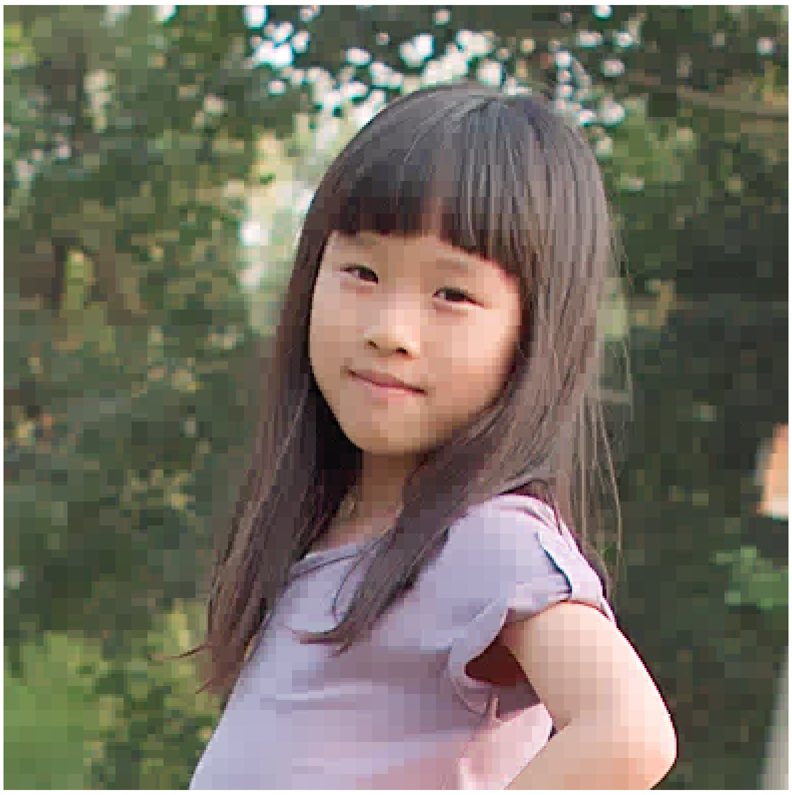
Code

```python
import cv2
import numpy as np

def haar8(arr):
    if arr.shape[0] != 8:
        print('haar shape wrong?')
        return arr
    # step 1
    arr1 = np.zeros(8)
    arr1[0] = np.mean(arr[0:2])
    arr1[1] = np.mean(arr[2:4])
    arr1[2] = np.mean(arr[4:6])
    arr1[3] = np.mean(arr[6:8])
    arr1[4] = arr[0] - arr1[0]
    arr1[5] = arr[2] - arr1[1]
    arr1[6] = arr[4] - arr1[2]
    arr1[7] = arr[6] - arr1[3]
    # step 2
    arr2 = np.zeros(4)
    arr2[0] = np.mean(arr1[0:2])
    arr2[1] = np.mean(arr1[2:4])
    arr2[2] = arr1[0] - arr2[0]
    arr2[3] = arr1[2] - arr2[1]
    # step 3
    arr3 = np.zeros(8)
    arr3[0] = np.mean(arr2[0:2])
    arr3[1] = arr2[0] - arr3[0]
    # concat result to arr3
    arr3[2:4] = arr2[2:4]
    arr3[4:8] = arr1[4:8]
    return arr3

def haar8_(arr):
    if arr.shape[0] != 8:
        print('haar shape wrong?')
        return arr
    arr1 = arr.copy()
    arr2 = arr.copy()
    # restore arr2
    arr2[0] = arr[0] + arr[1]
    arr2[1] = arr[0] - arr[1]
    # restore arr1
    arr1[0] = arr2[0] + arr2[2]
    arr1[1] = arr2[0] - arr2[2]
    arr1[2] = arr2[1] + arr2[3]
    arr1[3] = arr2[1] - arr2[3]
    # restore origin
    arr[0] = arr1[0] + arr1[4]
    arr[1] = arr1[0] - arr1[4]
    arr[2] = arr1[1] + arr1[5]
    arr[3] = arr1[1] - arr1[5]
    arr[4] = arr1[2] + arr1[6]
    arr[5] = arr1[2] - arr1[6]
    arr[6] = arr1[3] + arr1[7]
    arr[7] = arr1[3] - arr1[7]
    return arr

def trans8x8_0(img):
    if img.shape[0] != 8 or img.shape[1] != 8:
        print('transform size wrong?')
        return img
    # row trans
    for i in range(8):
        img[i] = haar8(img[i])
    # col trans
    for i in range(8):
        img[:, i] = haar8(img[:, i])
    return img

def trans8x8_1(img):
    if img.shape[0] != 8 or img.shape[1] != 8:
        print('transform size wrong?')
        return img
    # col reconstruct
    for i in range(8):
        img[:, i] = haar8_(img[:, i])
    # row reconstruct
    for i in range(8):
        img[i] = haar8_(img[i])
    return img

def transform(img, func):
    if img.shape[0] % 8 != 0 or img.shape[1] % 8 != 0:
        print('img size wrong?')
        return img
    for i in range(0, img.shape[0], 8):
        for j in range(0, img.shape[1], 8):
            img[i:i+8, j:j+8] = func(img[i:i+8, j:j+8])
    return img

if __name__ == '__main__':
    # print(haar8_(haar8(np.array([64,2,3,61,60,6,7,57]))))
    img = cv2.imread('original.bmp') * 1.
    delta = 5
    print('perform haar transform encode')
    for i in range(img.shape[-1] - 1, -1, -1):
        img[..., i] = transform(img[..., i], trans8x8_0)
    print('perform haar transform compress')
    img[(-delta <= img) & (img<= delta)] = 0
    print('perform haar transform decode')
    for i in range(img.shape[-1] - 1, -1, -1):
        img[..., i] = transform(img[..., i], trans8x8_1)
    cv2.imwrite('transform.png', img)
```

Result

- origin figure

- transformed figure ($\delta = 5$)



可以看到有明显的格子纹