ProvaVisão

André Meneghelli Vale - 4898948

25 DE JUNHO DE 2015

Q1. Para fazer esta questão, leia primeiro o artigo abaixo:

- `http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-85.pdf`
- Faça um resumo do artigo de acordo com as indicações que deixei no paca (artigos sobre como fazer um resumo).
- Implemente o método ELA (Error Level Analysis) em Python (apresente o **algoritmo** na prova e anexe o código em Python no arquivo zip).
- Teste seu algoritmo com as imagens que deixei no paca para este exercício. Quantas imagens seriam consideradas modificadas por esse método? Comente o resultado, comparando com a sua intuição.

- **RESUMO**:

A popularização de smartphones e a grande facilidade em se tomar imagens digitais acabou por tornar essas imagens como artefatos importantes na documentação de incidentes. Grandes portifólios de imagens podem ser acessados devido ao crescimento da internet, tornando comum o jornalismo informal.

Os avanços nas ferramentas de manipulação de imagens é tão grande, que atualmente torna-se bastante difícil a identificação de imagens manipuladas por analistas que não sejam peritos forenses especializados. Como consequência, essa nova fonte de documentação é bastante suscetível à falsificação.

Várias medidas estão sendo tomadas a fim de evitar problemas causados por imagens manipuladas, desde a necessidade judicial de verificação de autenticidade de imagens digitais à política de tolerância zero a imagens manipuladas por parte de mídias de comunicação profissionais. Essas medidas acabam consumindo grande quantidade de recursos, como o tempo dos peritos especializados, o dinheiro envolvido para manter métodos de detecção de fraudes ou, até mesmo, perda de dinheiro em seguros pagos indevidamente.

O artigo propõe a criação de um serviço web que possibilite a um usuário leigo a identificação de imagens modificadas, diminuindo assim a carga de trabalho dos peritos forenses e esporadicamente diminuir o prejuízo causado por fraudes. O desenvolvimento dessa ferramenta não pode ser patenteado devido à utilização de algoritmos extras, como a compactação de imagens com padrão JPEG, assim como tomar uma abordagem bastante intuitiva.

Os estudos para se desenvolvimento dessa ferramenta foca-se nos métodos mais comuns de manipulação de imagens: junção de imagens distntas e cópia de elementos para outras áreas da mesma imagem.

O método de compressão usado pelo JPEG consiste em três passos, que são transformação de cosseno discreta, quantização e codificação de entropia. A técnica de quantização acaba deixando ruídos que podem ser utilizados para provar se uma imagem é real, tornando a compactação um código de autenticação natural para detectar manipulações de imagens.

Duas técnicas foram desenvolvidas baseadas no passo de quantização: a primeira é a análise dos ruídos resultantes da compressão e a segunda é a análise do erro gerado no momento da compressão.

A técnica de análise de ruídos pode indicar se a imagem foi ou não manipulada, mas ainda é bastante bruta devido a maiores perdas nas regiões de alta frequência. Assim, esse algoritimo pode determinar se uma imagem foi comprimida.

Já o método que trata o erro gerado na compressão ELA (Error Level Analysis) se mostrou promissor, já que em imagens manipuladas é bastante provável que diferentes partes da imagem tenham níveis de erro distintos.

O algorítimo ELA funciona melhor com imagens compostas por imagens distintas, no entanto não é eficaz na análise de imagens com componentes copiados, já que esses possuem o mesmo nível de erro contido na imagem original.

Imagens com alta frequência apresentam níveis de erro elevados, embora não sejam adulteradas. A fim de solucionar esse inconveniente é gerada uma máscara de baixa frequência, aumentando a contribuição das áreas de baixa frequência para a imagem resultante. Depois de aplicar o filtro gaussiano, a imagem é normalizada para um tamanho máximo de 255 e novamente é limiarizada, a fim de reduzir ruídos.

Outra abordagem é rotular as áreas de alta e baixa frequência e identificar a sua intersecção. Geralmente isso é mais eficaz por ser visualmente mais agradável e por não ignorar regiões de alta frequência que tenham sido manipuladas.

A utilização de métricas numéricas junto com técnicas de aprendizado de máquina tem se mostrado bastante eficiente, potencializando o acerto para 85,54%.

- **Método ELA**:

```
def is_real(filename, quality=85):
    resaved = filename + '.resaved.jpg'
    im = Image.open(filename)
    im.save(resaved, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved)
    ela_im = ImageChops.difference(im, resaved_im)
    extrema = ela_im.getextrema()
    os.remove(resaved)
    return np.std([ex[1] for ex in extrema]) < 2
```

- **Testes**:

  Após analisar métricas numéricas dos resultados obtidos decidi que um bom algorítimo é o que restringe imagens verdadeiras à desvios padrões menores do que 2 para qualidade de 85% na compressão JPEG. É possível observar que o classificador automático falha muitas vezes desta maneira e a abordagem utilizando aprendizado de máquina talvez consiga trazer resultados mais satisfatórios.

  - 2000_snowballcat.jpg é falsa!
  - bikefail.jpg é falsa!
  - blacklion01.jpg é falsa!
  - bouncing_baby.jpg é falsa!
  - broken_road.jpg é falsa!
  - businahole.jpg é falsa!
  - cows_on_beach.jpg é falsa!
  - cursor.jpg é falsa!
  - daliatom.jpg é real!
  - dononwater.jpg é real!
  - eagles.jpg é falsa!
  - frozenvenice.jpg é falsa!
  - glass_butterfly.jpg é falsa!
  - goldfish_hitler.jpg é falsa!
  - hatfield.jpg é real!
  - hellephant.jpg é falsa!
  - hitlerbaby.jpg é real!
  - horseinahole01.jpg é falsa!
  - houseboat01.jpg é falsa!
  - iceberg.jpg é falsa!
  - jumping_giraffe.jpg é falsa!
  - kissing.jpg é real!
  - koala01.jpg é real!
  - leap.jpg é real!
  - magic_tap.jpg é falsa!
  - manitoba_security.jpg é real!
  - moonmelon01.jpg é falsa!
  - nikolatesla.jpg é real!
  - queensguard.jpg é real!
  - rainbow_tornado.jpg é falsa!
  - rocket_bike.jpg é falsa!
  - shark_roof.jpg é real!
  - sharkswim.jpg é falsa!
  - skiing_egypt.jpg é falsa!
  - skullrose.jpg é falsa!
  - spacechair.jpg é falsa!
  - tandembike.jpg é real!
  - tattooguy01.jpg é real!
  - tennis.jpg é real!
  - tentacle_bldg.jpg é falsa!
  - trafficlights.jpg é falsa!
  - tunnelface01.jpg é real!
  - verydeep.jpg é real!
  - vuitton.jpg é falsa!
  - whale.jpg é falsa!
  - wienerplane.jpg é falsa!

Q2. Esta questão refere-se à transformada de Fourier.

- Encontre a transformada de Fourier da função:

$$f(x) = \begin{cases} 7 & \text{if } -5 < x < 5 \\ 0 & \text{caso contrário} \end{cases}$$

- Encontre a transformada de Fourier da função $g(x) = f(x)\cos\omega_0 x$, sabendo que a transformada de Fourier de $f(x)$ é dada por $F(\omega)$
- Ache a inversa da transformada de Fourier de $G(\omega) = 20\frac{\sin 5\omega}{5\omega}e^{-3\omega i}$
- Calcule a DFT do sinal $f = \{1, 3, 5, 3, 1\}$

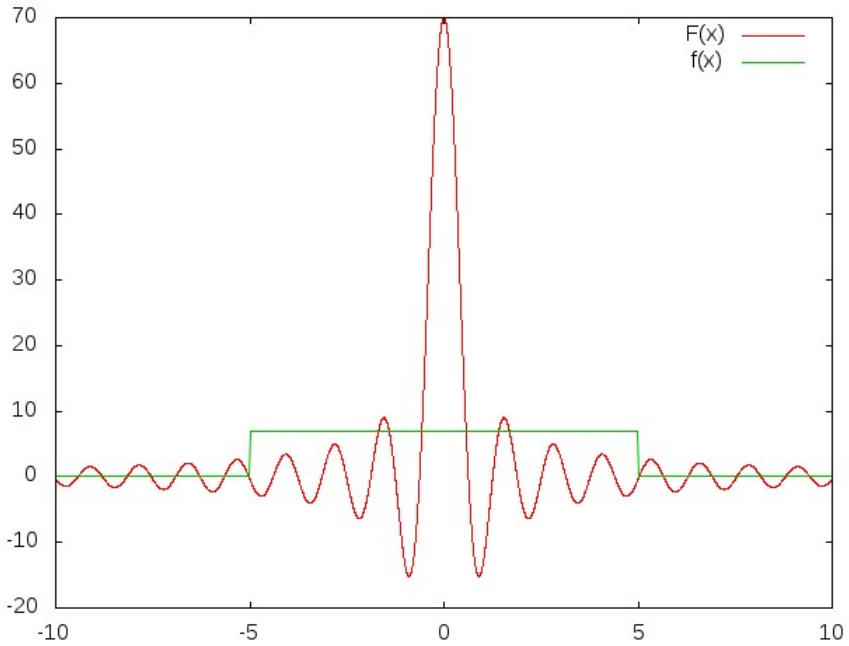
- **transformada de** $f(x)$:

$F(w) = AD\frac{\sin\frac{wD}{2}}{\frac{wD}{2}}$

$A = 7$

$D = 10$

$F(w) = 14\frac{\sin 5w}{w}$



- **transformada de** $g(x)$:

$$g(x) = \begin{cases} 7cos(w_0 x) & \text{if } -5 < x < 5 \\ 0 & \text{caso contrário} \end{cases}$$

$$\alpha = 2\pi w$$

$$G(x) = \int_{-\infty}^{\infty} g(x) e^{-i\alpha x} dx$$

$$G(x) = \int_{-\infty}^{-5} g(x) e^{-i\alpha x} dx + \int_{-5}^{5} g(x) e^{-i\alpha x} dx + \int_{5}^{\infty} g(x) e^{-i\alpha x} dx$$

$$G(x) = \int_{-5}^{5} 7cos(w_0 x) e^{-i\alpha x} dx$$

Substituindo pela identidade de Euler:

$$G(x) = 7\int_{-5}^{5} cos(w_0 x) cos(\alpha x) dx - 7i \int_{-5}^{5} cos(w_0 x) sen(\alpha x) dx$$

$$A(x) = \int_{-5}^{5} cos(w_0 x) cos(\alpha x) dx \qquad B(x) = i \int_{-5}^{5} cos(w_0 x) sen(\alpha x) dx$$

$$G(x) = 7(A(x) - B(x))$$

Calculando $A(x)$:

$$A(x) = \frac{1}{2} \left(\int_{-5}^{5} cos[(w_0 - \alpha)x] dx + \int_{-5}^{5} cos[(w_0 + \alpha)x] dx \right)$$

$$u = (w_0 - \alpha)x \qquad dx = \frac{du}{w_0 - \alpha}$$

$$v = (w_0 + \alpha)x \qquad dx = \frac{dv}{w_0 + \alpha}$$

$$A(x) = \frac{1}{2} \left(\frac{\int_{-5}^{5} cos(u) du}{w_0 - \alpha} + \frac{\int_{-5}^{5} cos(v) dv}{w_0 + \alpha} \right)$$

$$A(x) = \frac{1}{2} \left(\frac{sen[(w_0 - \alpha)x]}{w_0 - \alpha} + \frac{sen[(w_0 + \alpha)x]}{w_0 + \alpha} \right)$$

Calculando $B(x)$:

$$B(x) = \frac{1}{2}\left(\int_{-5}^{5} sen[(w_0+\alpha)x]dx - \int_{-5}^{5} sen[(w_0-\alpha)x]dx\right)$$

$$u = (w_0+\alpha)x \qquad dx = \frac{du}{w_0+\alpha}$$

$$v = (w_0-\alpha)x \qquad dx = \frac{dv}{w_0-\alpha}$$

$$B(x) = \frac{1}{2}\left(\frac{\int_{-5}^{5} sen(u)du}{w_0+\alpha} - \frac{\int_{-5}^{5} sen(v)dv}{w_0-\alpha}\right)$$

$$B(x) = \frac{1}{2}\left(\frac{cos[(w_0-\alpha)x]}{w_0-\alpha} - \frac{cos[(w_0+\alpha)x]}{w_0+\alpha}\right)$$

Então:

$$G(x) = 7(A(x) - B(x))$$

$$G(x) = 7\left(\frac{1}{2}\left(\frac{sen[(w_0-\alpha)x]}{w_0-\alpha} + \frac{sen[(w_0+\alpha)x]}{w_0+\alpha}\right) - \frac{1}{2}\left(\frac{cos[(w_0-\alpha)x]}{w_0-\alpha} - \frac{cos[(w_0+\alpha)x]}{w_0+\alpha}\right)\right)$$

- **inversa de** $G(w)$:

$$G^{-1}\left(G(w)\right) = \frac{1}{2\pi}\int_{-\infty}^{\infty} G(w)e^{iw_0 w}dw$$

$$g(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} 20\frac{sen5w}{5w}e^{-3wi}e^{iwx}$$

- **DFT do sinal** $f = \{1, 3, 5, 3, 1\}$

$$s(n) = \sum_{k=0}^{N-1} f(k) e^{\frac{2\pi i n k}{N}}$$

  - $\mathbf{n = 0}$:

$$s(0) = \sum_{k=0}^{4} f(k) e^{0} = 1 + 3 + 5 + 3 + 1 = 13$$

  - $\mathbf{n = 1}$:

$$s(1) = \sum_{k=0}^{4} f(k) e^{\frac{2\pi i k}{5}}$$

$$s(1) = 1e^{0} + 3e^{\frac{2\pi i}{5}} + 5e^{\frac{4\pi i}{5}} + 3e^{\frac{6\pi i}{5}} + 1e^{\frac{8\pi i}{5}} = -4.24 - 3.08i$$

  - $\mathbf{n = 2}$:

$$s(2) = \sum_{k=0}^{4} f(k) e^{\frac{4\pi i k}{5}}$$

$$s(2) = 1e^{0} + 3e^{\frac{4\pi i}{5}} + 5e^{\frac{8\pi i}{5}} + 3e^{\frac{12\pi i}{5}} + 1e^{\frac{16\pi i}{5}} = 0.24 + 0.73i$$

  - $\mathbf{n = 3}$:

$$s(3) = \sum_{k=0}^{4} f(k) e^{\frac{6\pi i k}{5}}$$

$$s(3) = 1e^{0} + 3e^{\frac{6\pi i}{5}} + 5e^{\frac{12\pi i}{5}} + 3e^{\frac{18\pi i}{5}} + 1e^{\frac{24\pi i}{5}} = 0.24 - 0.73i$$

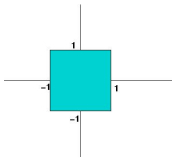
  - $\mathbf{n = 4}$:

$$s(4) = \sum_{k=0}^{4} f(k) e^{\frac{8\pi i k}{5}}$$

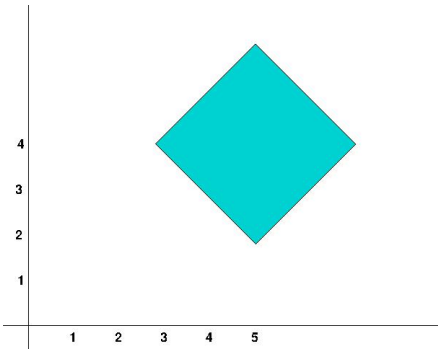
$$s(4) = 1e^{0} + 3e^{\frac{8\pi i}{5}} + 5e^{\frac{16\pi i}{5}} + 3e^{\frac{24\pi i}{5}} + 1e^{\frac{32\pi i}{5}} = -4.24 + 3.08i$$

Q3.

- Calcule (apresente os cálculos) dos descritores de Fourier das figuras 1a e 1b. Lembre-se que os pontos da borda do quadrado serão representados por pontos no plano de Argand-Gauss. Isto é, cada ponto no plano passa a ser um número complexo e a borda passa a ser um vetor de pontos complexos, como num sinal, mas com valores complexos.
- Para confirmar que seus cálculos estão corretos, implemente um programa em Python que receba como entrada um vetor de números complexos (que são as coordenadas das bordas) e retorne os descritores de Fourier do vetor de entrada. Você pode usar as funções fornecidas pela biblioteca NUMPY para facilitar a programação.
- Para reconstruir a curva, faça uma função que receba um vetor com os descritores de Fourier, um número N de descritores a serem usados e grafique os pontos num plano cartesiano (para fazer a mesma figura que fizemos nos slides das aulas 15 e 16.



(1a) Quadrado de lado 1



(1b) Quadrado de lado 3

- **Descritores de Fourrier de 1a e 1b**:

  Calculando 1a:

  f = [1+i, 1-i, -1-i, i-1]

  - **n = 0**:

$$s(0) = \sum_{k=0}^{3} f(k)e^{0} = (1+i) + (1-i) + (-1-i) + (i-1) = 0$$

  - **n = 1**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{2\pi ik}{4}}$$

$$s(1) = (1+i)e^{0} + (1-i)e^{\frac{2\pi i}{4}} + (-1-i)e^{\frac{4\pi i}{4}} + (-1+i)e^{\frac{6\pi i}{4}} = 0$$

  - **n = 2**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{4\pi ik}{4}}$$

$$s(1) = (1+i)e^{0} + (1-i)e^{\frac{4\pi i}{4}} + (-1-i)e^{\frac{8\pi i}{4}} + (-1+i)e^{\frac{12\pi i}{4}} = 0$$

  - **n = 1**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{6\pi ik}{4}}$$

$$s(1) = (1+i)e^{0} + (1-i)e^{\frac{6\pi i}{4}} + (-1-i)e^{\frac{6\pi i}{4}} + (-1+i)e^{\frac{6\pi i}{4}} = 4+4i$$

Calculando 1b:

f = [3+4i, 5+2i, 7+4i, 5+6i]

- **n = 0**:

$$s(0) = \sum_{k=0}^{3} f(k)e^{0} = (3+4i) + (5+2i) + (7+4i) + (5+6i) = 20+16i$$

- **n = 1**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{2\pi ik}{4}}$$

$$s(1) = (3+4i)e^{0} + (5+2i)e^{\frac{2\pi i}{4}} + (7+4i)e^{\frac{4\pi i}{4}} + (5+6i)e^{\frac{6\pi i}{4}} = -8$$

- **n = 2**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{4\pi ik}{4}}$$

$$s(1) = (3+4i)e^{0} + (5+2i)e^{\frac{4\pi i}{4}} + (7+4i)e^{\frac{8\pi i}{4}} + (5+6i)e^{\frac{12\pi i}{4}} = 0$$

- **n = 1**:

$$s(1) = \sum_{k=0}^{3} f(k)e^{\frac{6\pi ik}{4}}$$

$$s(1) = (3+4i)e^{0} + (5+2i)e^{\frac{6\pi i}{4}} + (7+4i)e^{\frac{6\pi i}{4}} + (5+6i)e^{\frac{6\pi i}{4}} = 0$$

- **Programa de cálculo dos descritores e função reconstrutora**: *fourier.py*
  - **Cálculo**:

    ```
    kterm $ python fourier.py -p "1+1j,1-1j,-1-1j,-1+1j"
    ```

  - **Reconstrução**:

    ```
    kterm $ python fourier.py --print -p "3+4j,5+2j,7+4j,5+6j"
    ```

```python
# coding=utf-8
import argparse
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon

def complex2tuple(points):
    # Funcao auxiliar para reconstruir a imagem.
    reals = []
    maxV = minV = minH = maxH = None
    for p in points:
        if maxV is None:
            maxV = minV = p.imag
            maxH = minH = p.real
        reals.append((p.real, p.imag))
        if p.real < minH:
            minH = p.real
        elif p.real > maxH:
            maxH = p.real
        if p.imag < minV:
            minV = p.imag
        elif p.imag > maxV:
            maxV = p.imag
    return reals, ([minH-1, maxH+1], [minV-1, maxV+1])

def reconstroi(descritores):
    # Funcao de reconstrucao.
    points = np.fft.ifft(descritores)
    rpoints, limits = complex2tuple(points)
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.add_patch(Polygon(rpoints, closed=True, fill=False, hatch='X'))
    ax.set_xlim(limits[0])
    ax.set_ylim(limits[1])
    plt.show()
    return points
```

```
def calcula_descritores(points):
    # Funcao de calculo dos descritores.
    return np.fft.fft(points)

def main():
    # Fluxo principal do programa.
    parser = argparse.ArgumentParser(
            description='Calculo dos descritores de Fourier.'
        )
    parser.add_argument('-p', '--points', metavar='"points"', type=str,
                        required=True, help='Vetor de pontos complexos.')
    parser.add_argument('--print', dest='out', action='store_true')
    parser.set_defaults(out=False)
    args = parser.parse_args()
    points = eval(args.points)
    descritores = calcula_descritores(points)
    if args.out:
        reconstroi(descritores)
    else:
        print descritores

if __name__ == "__main__":
    main()
```