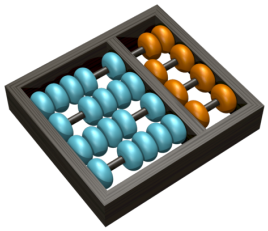
MC920 - Introdução ao Processamento de Imagens



Processamento de Imagens Digitais:
Comparação de Imagens
Plano de Bits
Entropia

FRANCISCO RAFAEL CARNEIRO RA 157888

UNICAMP
2017

1. Introdução

A análise e processamento de imagens digitais neste trabalho, consiste primeiramente em uma análise comparativa entre duas imagens, baseando-se em histogramas para cada canal de cor e na distância euclidiana. Já a segunda parte, parte-se para a extração dos planos de bits de uma imagem em níveis de cinza. E por fim, foi computado o valor da entropia de cada plano de bit.

2. Comparação de Imagens

2.1. Histogramas

Histogramas de imagens, são formas de representar a imagem disposta de acordo com a intensidade de cada pixel, ou seja, cada coluna de um histograma representa a quantidade de cada intensidade de pixel. Para esse problema, foi utilizada duas imagens, ambas coloridas. Portanto, para cada canal de cor (Red, Green, Blue(RGB)), foi aplicado os histogramas.
As imagens abaixo, foram as imagens utilizadas no processo comparativo:

(As imagens coloridas se encontram no diretório juntamente com os arquivos python e imagens)

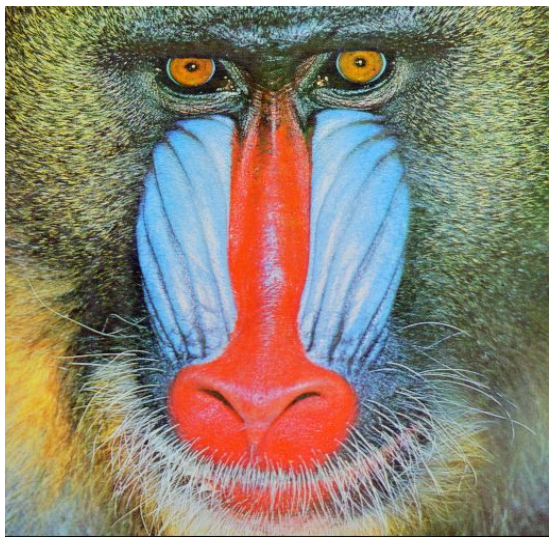


Figura 1: baboon.png

Figura 2: Imagem peppers.png

O histogramas foram gerados a partir de uma quantidade específica de bins. Cada bin, representa uma faixa de pixels. Por exemplo, uma imagem com 256 níveis de cinza, um

histograma com 256 bins, teria uma coluna para cada intensidade de pixel. Já um histograma com 128 bins, teria uma coluna para cada duas intensidade de pixel, sendo no primeiro bin os pixels 0 e 1, no segundo bin 2 e 3 e assim sucessivamente.
O algoritmo utilizado na geração dos histogramas, consistia no código:

```
def bin_image(img, m, n):
    r = np.zeros((256))
    g = np.zeros((256))
    b = np.zeros((256))
    for i in range (m):
        for j in range (n):
            r[img[i][j][0]] = r[img[i][j][0]] + 1
            g[img[i][j][1]] = g[img[i][j][1]] + 1
            b[img[i][j][2]] = b[img[i][j][2]] + 1
     return r, g, b
```

Era criado um vetor que continha todos os possíveis valores de pixels e nesses era adicionados os valores de cada pixel e por fim, retorna os vetores para cada canal de cor.. Lembrando que, nesse caso, essa função funciona somente para imagens RGB.

O Histograma gerado, contém, nesse caso, 256 bins (tamanho do vetor), assim, criou-se uma função que retorna um vetor com a quantidade de bins desejada.

2.2. Distância Euclidiana.

Outra métrica para comparação de imagens, foi a distância euclidiana. Essa foi utilizada para cada nível do histograma. A distância euclidiana é método comparativo que indica o quão diferente são as imagens à nível de cor. Para um melhor cálculo, foi feita a média dos histogramas de cada canal de cor de uma imagem colorida e então foi calculada a distância entre as duas imagens.

O algoritmo usado, foi:

```
def dist_euclidiana(hist1, hist2):
    soma = 0;
    tam = len(hist1)
    for i in range (tam):
            soma = soma + ((hist1[i]- hist2[i])**2)

    return math.sqrt(soma)
```

Onde era calculado o valor da distância, percorrendo os histogramas e fazendo o quadrado das diferenças e por fim, retornando a raiz quadrada dos valores.

2.3. Comparação para 4 bins.

Foi feito os histogramas para cada 4 bins, assim temos:

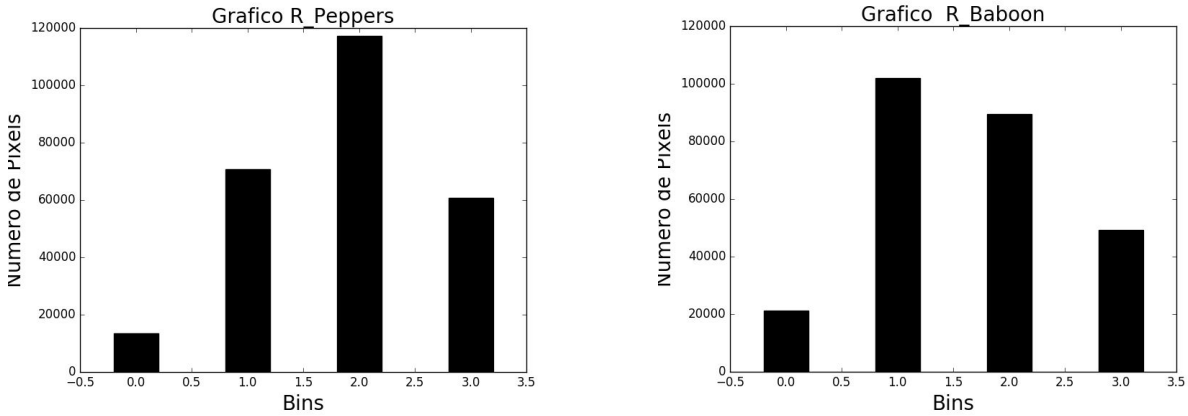


Figura 3: Histogramas 4 bins para o canal R

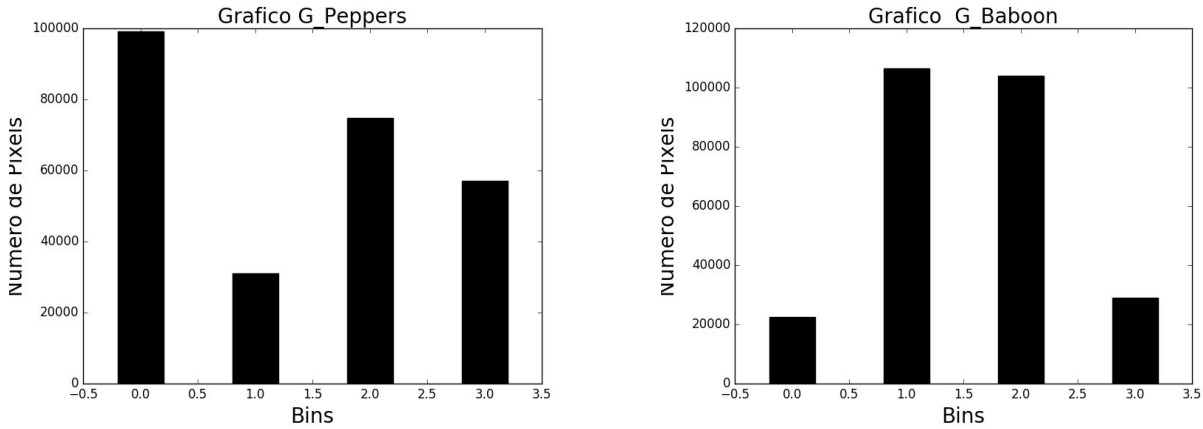


Figura 4: Histogramas 4 bins para o canal G

Figura 5: Histogramas 4 bins para o canal B

Com isso, foi feita a média da distância Euclidiana entre os 3 canais.
Essa por sua vez teve valor de: 0.3115
Um valor menor da distância significa que as imagens são parecidas só que no âmbito da distribuição de cores, ou seja, seus histogramas são parecidos (no caso a média).

2.4. Comparação para 32 bins.

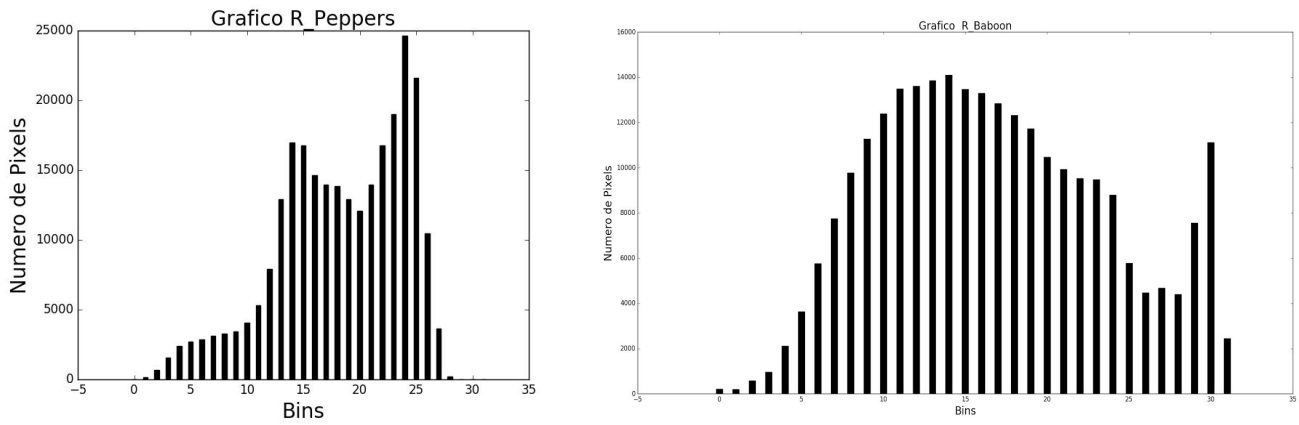


Figura 6: Histogramas 32 bins para o canal R

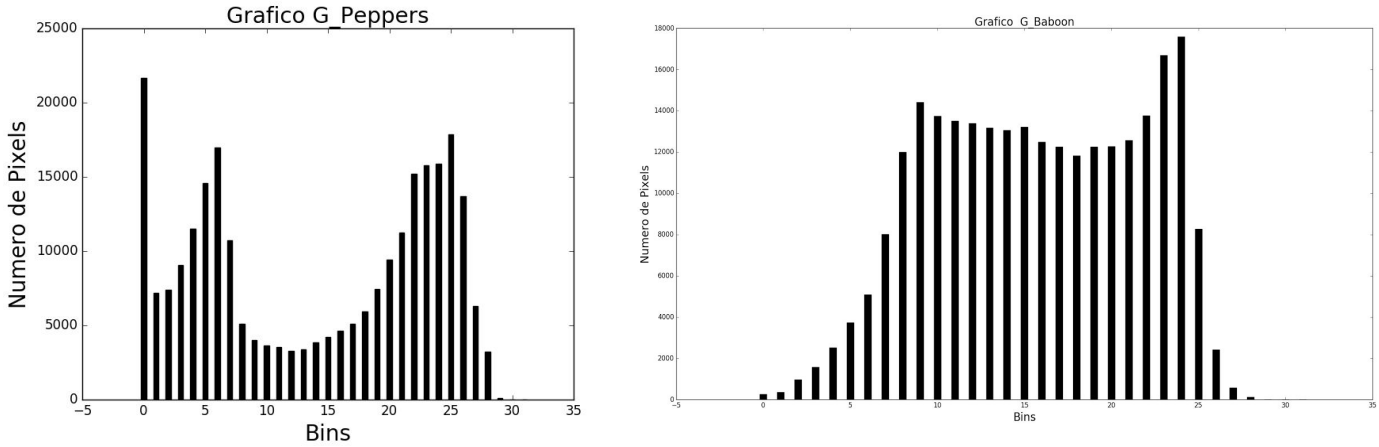


Figura 7: Histogramas 32 bins para o canal G

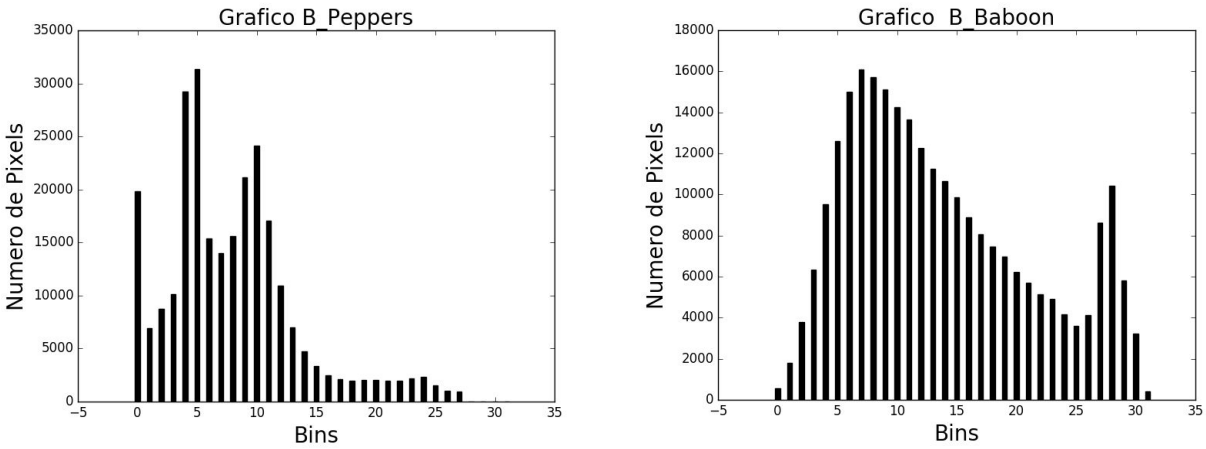
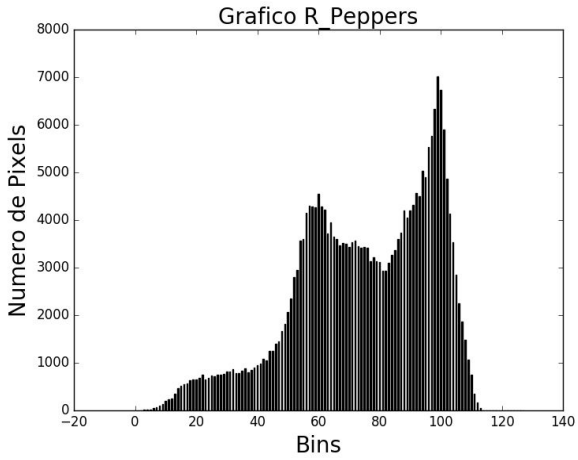


Figura 8: Histogramas 32 bins para o canal B

O valor da distância euclidiana obtida entre os histogramas com 32 bins foi: 0.1576
Ou seja, vimos que se aumentar a quantidade da distribuição nos histogramas, a distância diminui, ou seja, as imagens tendem a se parecer mais no âmbito da distribuição de intensidades.

2.5. Comparação para 128 bins.



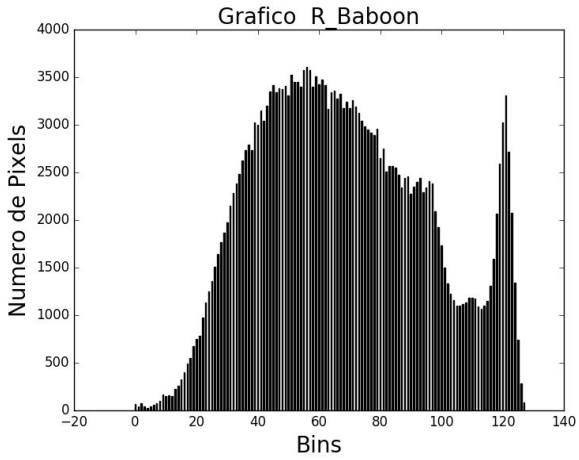
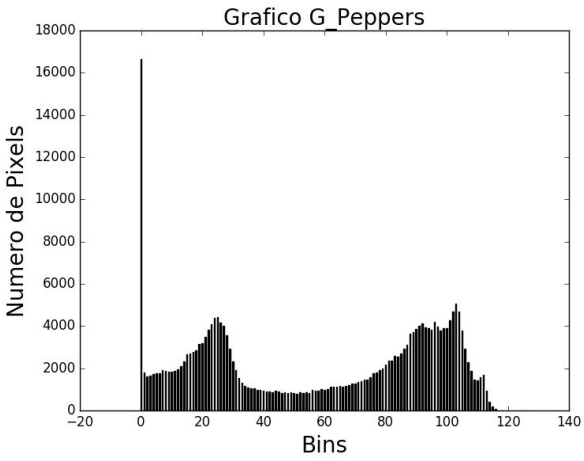


Figura 9: Histogramas 128 bins para o canal R



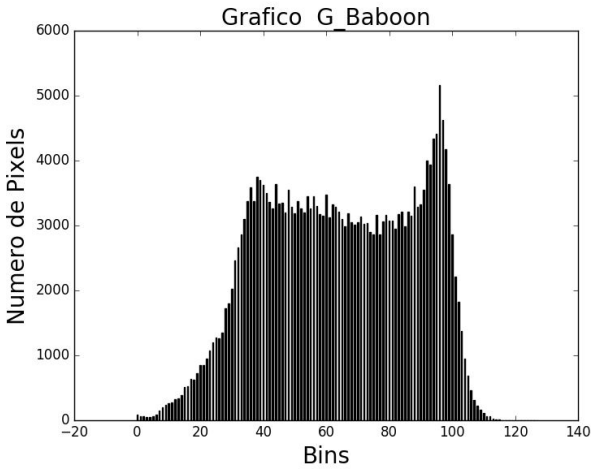
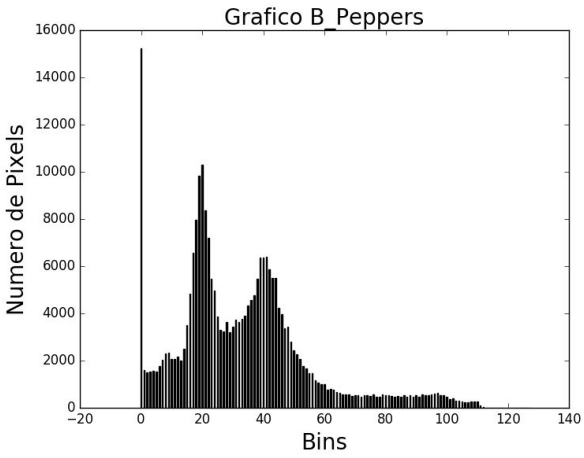


Figura 10: Histogramas 128 bins para o canal G



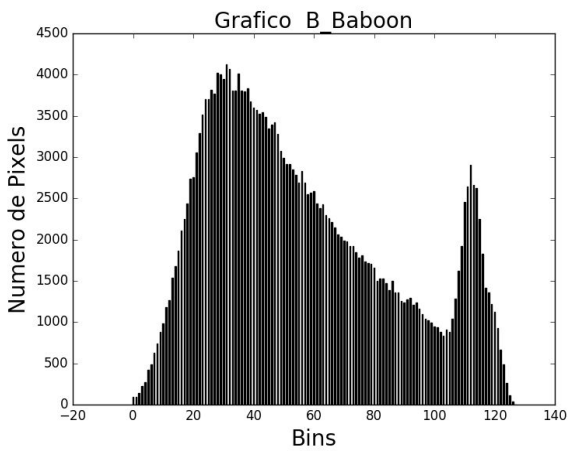


Figura 11: Histogramas 128 bins para o canal B

O valor da distância euclidiana foi de: 0.0888.

2.6. Comparação para 256 bins.

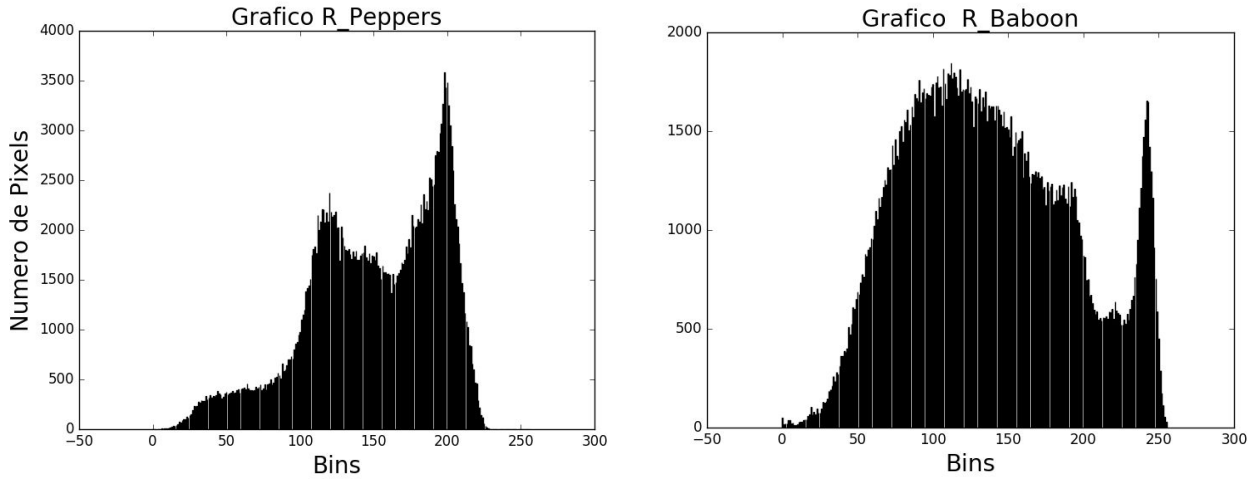


Figura 11: Histogramas 256 bins para o canal R

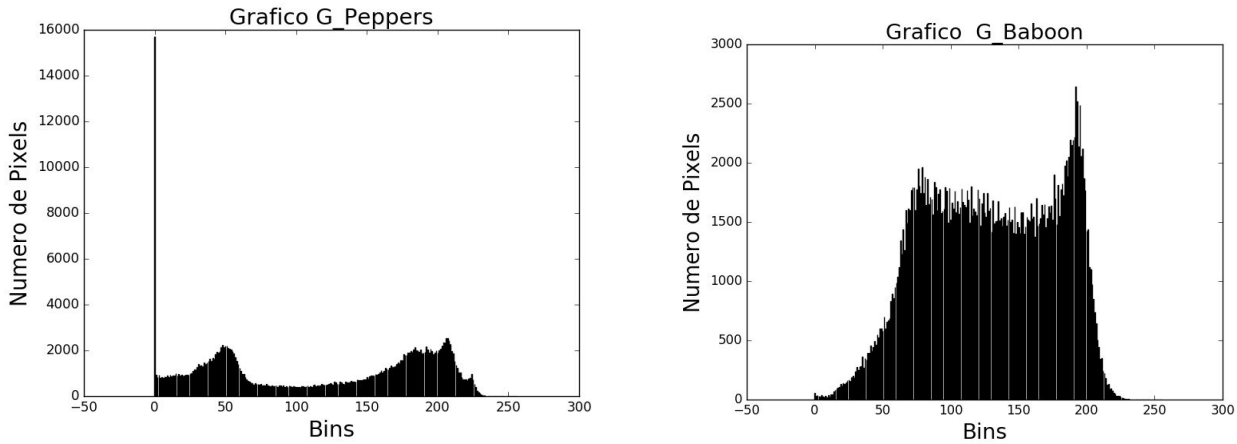


Figura 12: Histogramas 256 bins para o canal G

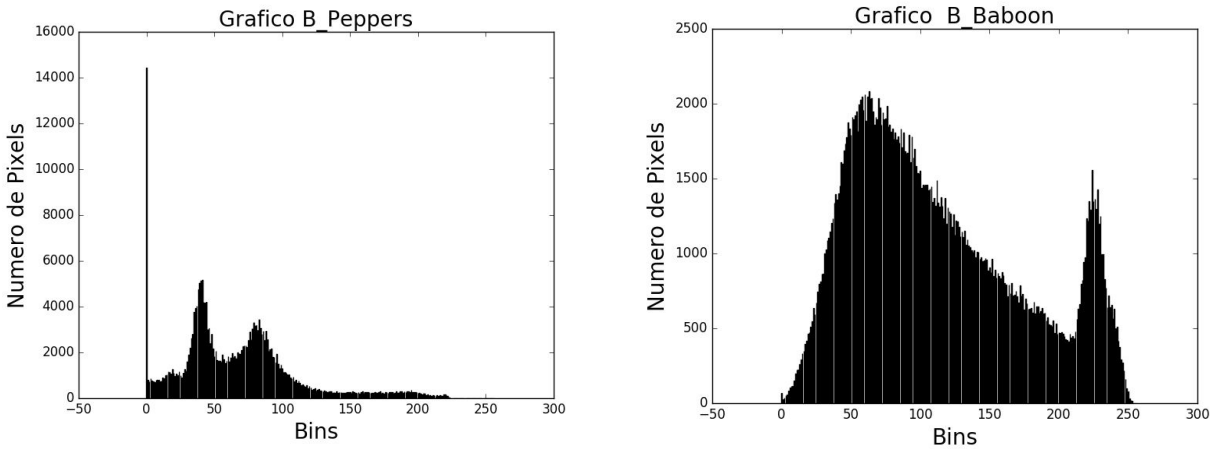


Figura 13: Histogramas 256 bins para o canal B

Por fim, o valor da distância para 256 bins foi de: 0.0694.
Percebemos assim, que as imagens possuem uma distribuição parecida, uma vez que, quanto mais foi discretizada (aumento no número de bins) menor foi a distância euclidiana. Apesar de poder afirmar que as imagens possuem um histograma parecido, não podemos afirmar que as imagens são iguais e sim que elas têm uma distribuição de cor parecidas - nos níveis de intensidade.

2.7. Plano de Bits

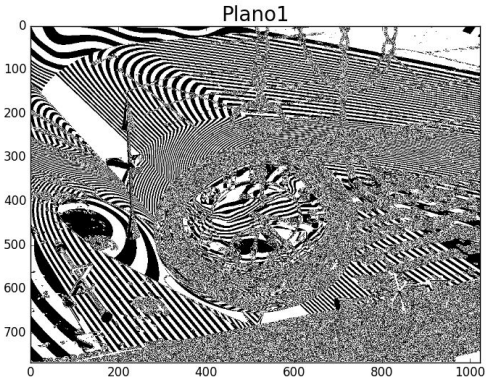
A segunda parte consiste na extração dos planos de bits de imagens em níveis de cinza. Foi feita uma conversão de uma imagem colorida para uma imagem em níveis de cinza. A extração dos planos de bits, tratava-se de remover os bits de uma imagem com 8 bits. Por exemplo, suponha-se que um bit de uma imagem de oito bits seja 00001111, logo, o plano 8, mais significativo (da direita para esquerda) teria 0 na intensidade desse bit, o plano 7, 0, o plano 1 teria intensidade 1 (255). Ou seja, seriam formadas imagens binárias.

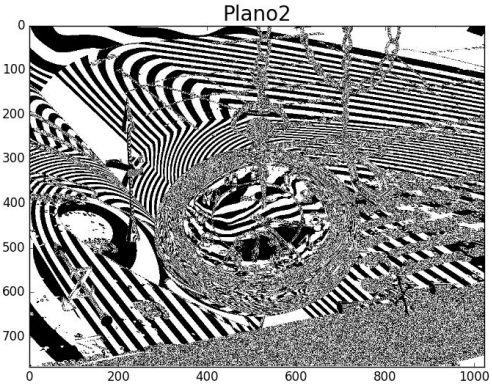
O algoritmo que no qual gerou os planos foi:

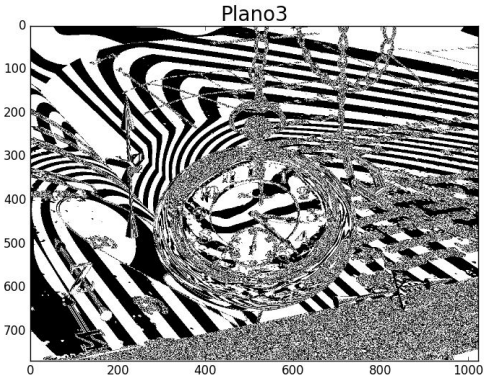
```
def planos(gray_iamge, M, N, plan):
    planbin = np.zeros(shape = (M,N))
    for i in range(M):
        for j in range (N):
            plan_bin = bin(gray_image[i][j])[2:].zfill(8)
            planbin[i][j] = plan_bin[8-plan]

     return planbin
```

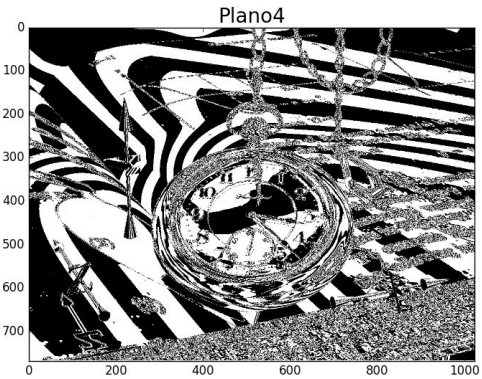
No caso dessa função, para cada pixel da matriz base, criava-se um vetor que continha a intensidade do pixel em binário, sendo a posição 0 a parte menos significativa e a posição máxima, o bit mais significativo.
Fora utilizada a função *".zfill(8)",* que normalizava todos os valores com 8 bits.
Assim, retornava o plano (a matriz/imagem).

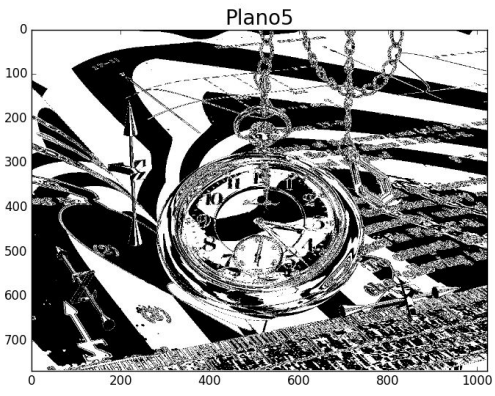
Partindo da imagem peppers e watch, obteve:

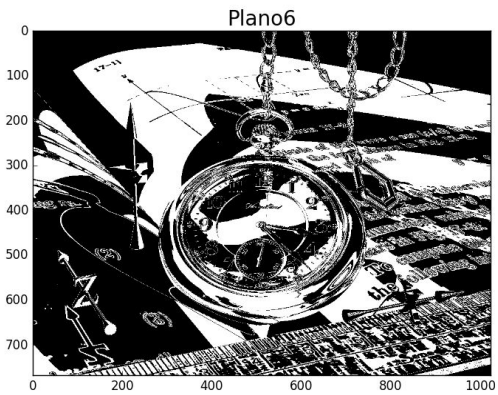

Plano1
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

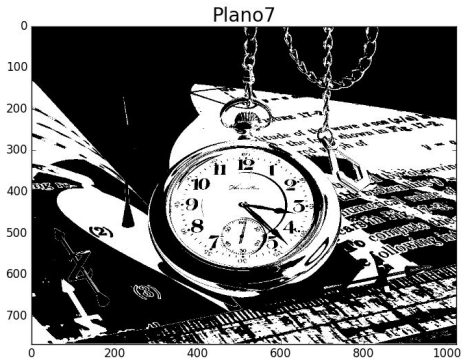

Plano2
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


Plano3
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


Plano4
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


Plano5
0
100
200
300
400
500
600
700
0
200
400
600
800
1000


Plano6
0
100
200
300
400
500
600
700
0
200
400
600
800
1000



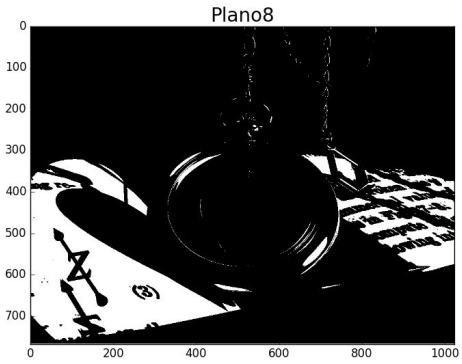
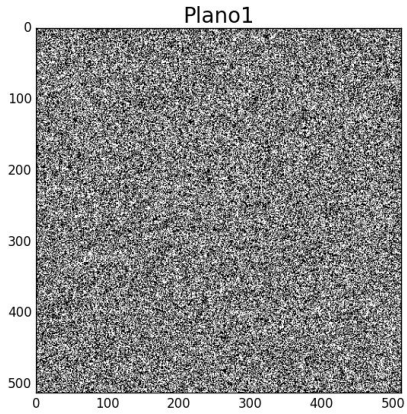
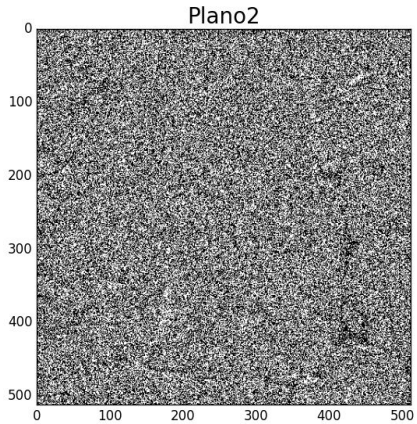
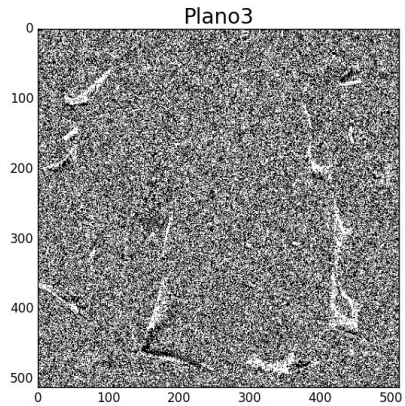
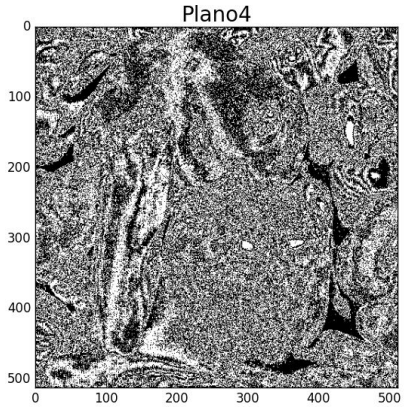


Figura 14: Planos da imagem watch.png









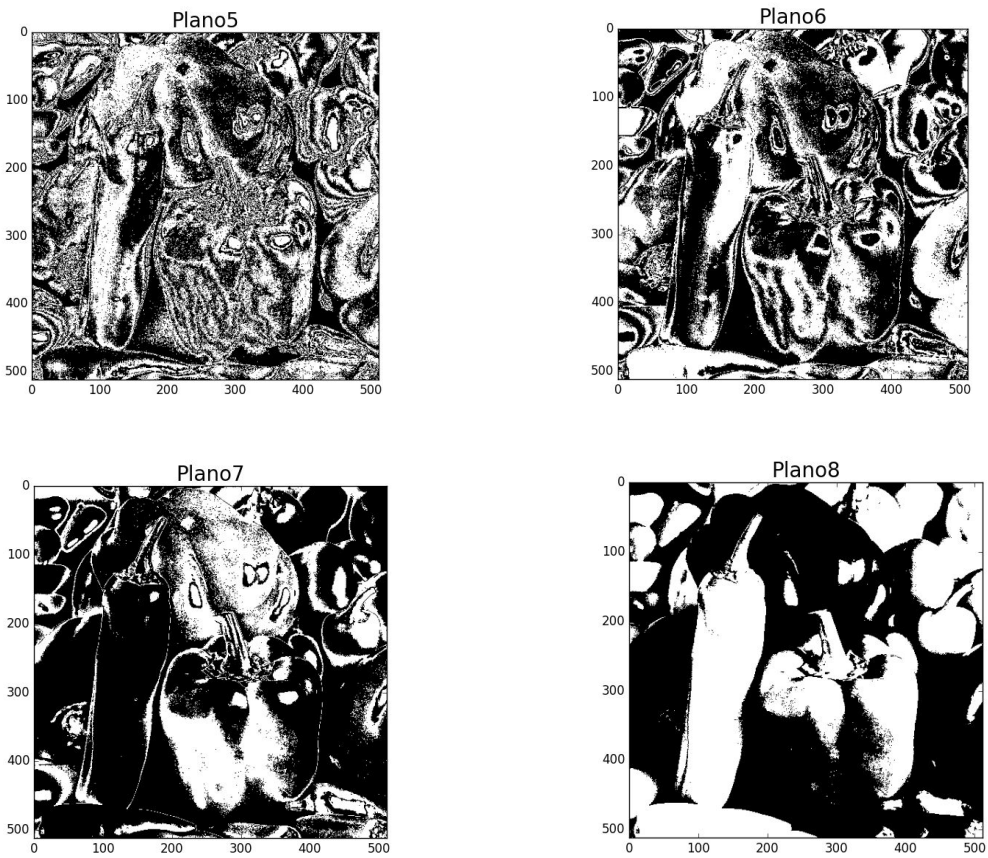


Figura 15: Planos da imagem peppers.png

Podemos observar que nos planos dos bits menos significativos não dá para obter nenhuma informação da imagem visualmente. A medida com que a significância dos bits cresce, a quantidade de informação aumenta, podendo ter ideia do que se trata a image, como por exemplo no plano 8 da imagem peppers.png.

2.8. Entropia

Após gerar os planos binários de bits de cada imagem, foi calculado os valores da entropia de cada um dos planos.
A entropia, consiste em um valor que leva a informação de um canal ou de uma fonte.

$$H = -\sum_{i=0}^{L-1} p_i \log_2 p_i$$

Onde *pi* é a probabilidade de um pixel assumir o valor i.
Em uma imagem binária, o valor máximo de entropia seria 1, uma vez que as duas cores tivessem a mesma quantidade de intensidade. O menor valor é 0, é quando existe somente uma intensidade na imagem.
O cálculo foi:

```
def entropia(plan_img, M, N):
    num_pixels = float(M*N)
    num_um = np.count_nonzero(plan_img)
    num_zero = num_pixels - num_um
    p_um = num_um/num_pixels
    p_zero = num_zero/num_pixels
    print p_um + p_zero
    entr = -1*(p_zero*np.log2(p_zero) + p_um*np.log2(p_um))

    return entr
```

Como as imagens que seriam utilizadas para o cálculo da entropia eram binárias, foi utilizada uma função da biblioteca *numpy* na qual contava todos os valores diferentes de 0 (bits 1) e assim, foi feita a diferença entre o total e a quantidade de 1 para achar a quantidade de 0. Assim, achou as probabilidades de cada um e fez-se o cálculo da entropia.

Assim, para as imagens peppers e watch obteve:

Imagem Watch.png
Entropia do plano 1 : 0.998483700728
Entropia do plano 2 : 0.998412533175
Entropia do plano 3 : 0.997454424913
Entropia do plano 4 : 0.997817000041
Entropia do plano 5 : 0.999997874359
Entropia do plano 6 : 0.971652984314
Entropia do plano 7 : 0.950433723928
Entropia do plano 8 : 0.623017944941

Percebe-se que nos primeiros planos a entropia tende ao valor máximo, ou seja, a quantidade de pixels 0 e 255 são muito similares, diferentemente no plano 8, uma vez que a quantidade de 255, como pode-se observar, é muito maior.

Imagem peppers.png
Entropia do plano 1 : 0.999999882056
Entropia do plano 2 : 0.999989670267
Entropia do plano 3 : 0.999912447639
Entropia do plano 4 : 0.999387849834
Entropia do plano 5 : 0.996637051907
Entropia do plano 6 : 0.992241675421
Entropia do plano 7 : 0.90164323232
Entropia do plano 8 : 0.986608674035

Na imagem peppers, pode-se observar também, que os valores são muito próximos à entropia máxima, uma vez que, a quantidade de pixels por canal (0 e 255) é muito próxima.

2.9 Considerações finais e Conclusão.

Os algoritmos utilizados foram em sua maioria sem a adição de funções prontas em algumas bibliotecas da linguagem python. Isso ocorreu uma vez que, ao meu entendimento, construir tudo ajudaria mais na fixação do conteúdo e melhor aprendizado sobre, como por exemplo dos histogramas, onde existe uma função da biblioteca *OpenCV* pronta para fazê-la.
Algumas limitações existem, como por exemplo, os histogramas funcionam somente para imagens com 3 canais de cores, no caso, RGB. Já a entropia é calculado somente para imagens binárias.

Pode-se concluir que, o trabalho trouxe uma boa visão na questão da representatividade de imagens - como no caso dos histogramas, onde pode ter uma análise mais franca das intensidades de uma image. Por outro lado, em termos comparativos, vimos que comparar somente histogramas é muito pouco para uma análise mais profunda sem levar em outra outros parâmetros.
Os planos de bits evidenciaram a composição de uma imagem, onde pode-se ver que os bits menos significativos não influenciam tanto no formato da imagem, em contrapartida dos bits mais significativos.

E por fim, a entropia nos da uma ideia de uma proximidade de cada canal de intensidade, uma, vez que, quanto o mais próxima do valor máximo, mais próximas estão as quantidades de pixels dos canais.