

Imagens Híbridas

Instruções

O código foi desenvolvido utilizando a linguagem de programação GNU Octave, onde no diretório do arquivo disponibilizado "*Color_Calibration_Histogram/code*" encontram -se todo o desenvolvimento da atividade solicitada, na qual em cada trecho do código existem comentários sobre seu funcionamento. Também neste mesmo arquivo foram inseridos as imagens utilizadas para a execução do algoritmo sendo elas: "*img1.png*", "*img1_patch.png*", "*img2.png*" e "*img2_patch.png*", podendo todas serem encontradas no diretório "*Color_Calibration_Histogram/data*".

Para a execução desta atividade é necessário a instalação dos pacotes "*Image*" e "*optim*" do GNU Octave, uma vez realizado a instalação do pacote, basta carregar o projeto em uma plataforma com suporte ao octave, executar o projeto definido como "*Code.m*" e definir o diretório padrão como sendo "*Color_Calibration_Histogram*". Após a execução do código é necessário aguardar um determinado tempo, sendo este tempo proporcional ao valor numérico da variável "*temp*" que representa o número de vezes que o código irá se repetir, em média esta variável com valor de 5 levou de 8 a 10 minutos. Por padrão a variável "*temp*" será iniciada com o valor de 15.

Detalhes de Implementação

Basicamente todo o código principal encontra -se dentro do arquivo definido como "*Code.m*", além deste arquivo neste diretório encontram -se outros dois arquivos sendo eles: "*D.m*" e "*Sistema.m*", sendo o primeiro arquivo referente à função que calcula a distância entre pontos por meio da distância euclidiana, que recebe como argumento dois histogramas referentes as imagens e retorna a soma total da distância entre os pontos. A seguir é apresentado seguinte trecho de código referente ao arquivo "*Sistema.m*":

```

1 function dis = D(u,v)
2   'inicializacao de variaveis
3   dis = 0;
4   [x,y] = size(u);
5
6   for i= (1:x)
7     'acumulo da diferenca ao quadrado de dois pontos
8     dis = dis + (u(x) - v(x))^2;
9   endfor
10 endfunction

```

O arquivo definido como "*Sistema.m*" armazena a função que representa um polinômio de segundo grau, apresentada como: $ax^2+bx+c = y$, sendo assim, esta função recebe como

argumentos os valores $x1, y1, x2, y2, x3, y3$, representando três pontos que em combinação irão tentar encontrar a melhor curva para o mapeamento de ajuste de cores. Esta função retorna como valor um vetor contendo três valores, representando as três soluções do sistema para os três pontos. A seguir é apresentado o trecho de código referente ao arquivo "*Sistema.m*":

```

1 function res = Sistema(x1,y1,x2,y2,x3,y3)
2     A = [x1^2, x1, 1; x2^2, x2 , 1; x3^2, x3, 1];
3     B = [y1;y2;y3];
4
5     res = A\B;
6 endfunction

```

Para o arquivo definido como "*Code.m*", o processo de inicialização de variáveis é apresentado através do seguinte trecho de código:

```

1 pkg load image;      #load packages
2 pkg load optim;     #load packages
3
4 image01 = imread("data/img1_patch.png");   #load images
5 image02 = imread("data/img2_patch.png");   #load images
6
7 figure('name','Imagen Inicial','numbertitle','off'),
8     imshow(image01);                      #apresentation of original
9     images
10    figure('name','Imagen Para Comparacao','numbertitle','off'
11        ), imshow(image02);      #apresentation of original
12        images
13
14    image01_out_R = zeros(1,256);    #output channel red
15    image01_out_G = zeros(1,256);    #output channel green
16    iamge01_out_B = zeros(1,256);    #output channel blue
17    [size_X,size_Y,size_Z] = size(image01);    #dimensions
18    of image
19    x1 = 1; y1 = 1;                  #initial
20    point to P1
21    x2 = 50; y2 = 45;              #initial
22    point to P2
23    x3 = 120; y3 = 100;            #initial
24    point to P3
25    new_vetor = zeros(1,256,"uint8"); #map color
26    1 to 256
27    temp=5;                      #initial
28    temperature
29    image_final = image01;          #create
30    var to final iterate image
31    best_image = zeros(size_X,size_Y,size_Z);    #create

```

```

1   var to vest image
21 diff_red = 0;                                #diff
22     perl channel (Red)
22 diff_green = 0;                               #diff
23     perl channel (Green)
23 diff_blue = 0;                                #diff
24     perl channel (Blue)
24 best_difR = inf(1);                           #var best
25       difference init with +inifinite Red
25 best_difG = inf(1);                           #var best
26       difference init with +inifinite Green
26 best_difB = inf(1);                           #var best
27       difference init with +inifinite Blue
27 new_dif = 0;                                  #diff in
28   loop

```

Para o processo de reconhecimento simulado, tal algoritmo só é finalizado quando a temperatura definida inicialmente atingir o valor zero, uma vez atingido tal valor, o algoritmo deverá selecionar a imagem que apresentou melhores resultados dentre todas as outras imagens geradas, utilizando a função de cálculo da distância para saber a proximidade entre a imagem que foi gerada em relação a imagem utilizada como comparação, armazenando assim os melhores resultados de cada decremento da temperatura, sendo a imagem utilizada para comparação armazenada pela variável "*image02*".

Para cada redução unitária da variável "*temp*", é percorrido dentre os três canais RGB (Red, Green e Blue) das imagens originais, buscando a diferença entre seus histogramas, uma vez tendo a diferença entre os histogramas das imagens originais separadas por canal é selecionado três pontos de forma aleatória, na qual esses pontos são utilizados para minimizar a diferença entre as cores da imagem original utilizada como comparação, sendo tais pontos incrementados de forma aleatória. Sendo assim tal exemplificação é apresentada através do trecho de código a seguir:

```

1 while (temp > 0)
2   printf("Temperatura: %d ... \n", temp);
3   i=1;                                #
4     iterate in channels
4   while (i <= 3)                      #
5     repeat to all channels RGB
5     image01_histogram=(hist((image01(:,:,i)),256));    #
6       histogram per channel 01
6     image02_histogram=(hist((image02(:,:,i)),256));    #
7       histogram per channel 02
7     dif = D(image01_histogram,image02_histogram);      #
8       difference between histograms
8     new_dif = 0;                                #
9       difference on loop between images

```

```

9    saida = zeros(size_X,size_Y);          #output
10   at final iteration
11
12   while (new_dif <= dif)
13     #print iteration
14     printf("Iniciando Iteracao... [%d,%d] [%d,%d] [%d,%d]\n",x1,y1,x2,y2,x3,y3);
15
16     #solution results for all three points
17     solution = Sistema(x1,y1,x2,y2,x3,y3);      #return
18     three points 2function ax^2+bx+c=y
19
20     #create color map
21     for aux= (0:255)
22       new_vetor(aux+1) = (solution(1)*(aux^2)) + (
23         solution(2)*aux) + (solution(3));      #a b c
24   endfor
25
26   #plot (0:255,new_vetor); #plot color map
27
28   for aux_i= (1:size_X)
29     for aux_j= (1:size_Y)
30       val_orig = (image01(aux_i,aux_j,i));
31       val_new = new_vetor(val_orig+1);
32       saida(aux_i,aux_j) = val_new;
33     endfor
34   endfor
35
36   poinSorted = randi(3);           #point to sort (p1,p2,
37   p3)
38   poinSorted_aux = randi(2);    #axis (x,y)
39   switch(poinSorted)
40     case 1
41       if ((x1 <= 80) || (y1 <= 80))  #p1 interval
42         (1-80)
43         if (poinSorted_aux == 1)
44           x1+=5;
45         else
46           y1+=5;
47         endif
48       else
49         if (poinSorted_aux == 1)
50           x1 = randi(85);
51         else
52           y1 = randi(85);
53         endif
54       endif
55     endswitch
56
57   for aux= (1:255)
58     new_vetor(aux+1) = (solution(1)*(aux^2)) + (
59       solution(2)*aux) + (solution(3));      #a b c
60   endfor
61
62   #plot (0:255,new_vetor); #plot color map
63
64   for aux_i= (1:size_X)
65     for aux_j= (1:size_Y)
66       val_orig = (image01(aux_i,aux_j,i));
67       val_new = new_vetor(val_orig+1);
68       saida(aux_i,aux_j) = val_new;
69     endfor
70   endfor
71
72   poinSorted = randi(3);           #point to sort (p1,p2,
73   p3)
74   poinSorted_aux = randi(2);    #axis (x,y)
75   switch(poinSorted)
76     case 1
77       if ((x1 <= 80) || (y1 <= 80))  #p1 interval
78         (1-80)
79         if (poinSorted_aux == 1)
80           x1+=5;
81         else
82           y1+=5;
83         endif
84       else
85         if (poinSorted_aux == 1)
86           x1 = randi(85);
87         else
88           y1 = randi(85);
89         endif
90       endif
91     endswitch
92
93   for aux= (1:255)
94     new_vetor(aux+1) = (solution(1)*(aux^2)) + (
95       solution(2)*aux) + (solution(3));      #a b c
96   endfor
97
98   #plot (0:255,new_vetor); #plot color map
99
100  for aux_i= (1:size_X)
101    for aux_j= (1:size_Y)
102      val_orig = (image01(aux_i,aux_j,i));
103      val_new = new_vetor(val_orig+1);
104      saida(aux_i,aux_j) = val_new;
105    endfor
106  endfor
107
108  poinSorted = randi(3);           #point to sort (p1,p2,
109  p3)
110  poinSorted_aux = randi(2);    #axis (x,y)
111  switch(poinSorted)
112    case 1
113      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
114        (1-80)
115        if (poinSorted_aux == 1)
116          x1+=5;
117        else
118          y1+=5;
119        endif
120      else
121        if (poinSorted_aux == 1)
122          x1 = randi(85);
123        else
124          y1 = randi(85);
125        endif
126      endif
127    endswitch
128
129  for aux= (1:255)
130    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
131      solution(2)*aux) + (solution(3));      #a b c
132  endfor
133
134  #plot (0:255,new_vetor); #plot color map
135
136  for aux_i= (1:size_X)
137    for aux_j= (1:size_Y)
138      val_orig = (image01(aux_i,aux_j,i));
139      val_new = new_vetor(val_orig+1);
140      saida(aux_i,aux_j) = val_new;
141    endfor
142  endfor
143
144  poinSorted = randi(3);           #point to sort (p1,p2,
145  p3)
146  poinSorted_aux = randi(2);    #axis (x,y)
147  switch(poinSorted)
148    case 1
149      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
150        (1-80)
151        if (poinSorted_aux == 1)
152          x1+=5;
153        else
154          y1+=5;
155        endif
156      else
157        if (poinSorted_aux == 1)
158          x1 = randi(85);
159        else
160          y1 = randi(85);
161        endif
162      endif
163    endswitch
164
165  for aux= (1:255)
166    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
167      solution(2)*aux) + (solution(3));      #a b c
168  endfor
169
170  #plot (0:255,new_vetor); #plot color map
171
172  for aux_i= (1:size_X)
173    for aux_j= (1:size_Y)
174      val_orig = (image01(aux_i,aux_j,i));
175      val_new = new_vetor(val_orig+1);
176      saida(aux_i,aux_j) = val_new;
177    endfor
178  endfor
179
180  poinSorted = randi(3);           #point to sort (p1,p2,
181  p3)
182  poinSorted_aux = randi(2);    #axis (x,y)
183  switch(poinSorted)
184    case 1
185      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
186        (1-80)
187        if (poinSorted_aux == 1)
188          x1+=5;
189        else
190          y1+=5;
191        endif
192      else
193        if (poinSorted_aux == 1)
194          x1 = randi(85);
195        else
196          y1 = randi(85);
197        endif
198      endif
199    endswitch
200
201  for aux= (1:255)
202    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
203      solution(2)*aux) + (solution(3));      #a b c
204  endfor
205
206  #plot (0:255,new_vetor); #plot color map
207
208  for aux_i= (1:size_X)
209    for aux_j= (1:size_Y)
210      val_orig = (image01(aux_i,aux_j,i));
211      val_new = new_vetor(val_orig+1);
212      saida(aux_i,aux_j) = val_new;
213    endfor
214  endfor
215
216  poinSorted = randi(3);           #point to sort (p1,p2,
217  p3)
218  poinSorted_aux = randi(2);    #axis (x,y)
219  switch(poinSorted)
220    case 1
221      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
222        (1-80)
223        if (poinSorted_aux == 1)
224          x1+=5;
225        else
226          y1+=5;
227        endif
228      else
229        if (poinSorted_aux == 1)
230          x1 = randi(85);
231        else
232          y1 = randi(85);
233        endif
234      endif
235    endswitch
236
237  for aux= (1:255)
238    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
239      solution(2)*aux) + (solution(3));      #a b c
240  endfor
241
242  #plot (0:255,new_vetor); #plot color map
243
244  for aux_i= (1:size_X)
245    for aux_j= (1:size_Y)
246      val_orig = (image01(aux_i,aux_j,i));
247      val_new = new_vetor(val_orig+1);
248      saida(aux_i,aux_j) = val_new;
249    endfor
250  endfor
251
252  poinSorted = randi(3);           #point to sort (p1,p2,
253  p3)
254  poinSorted_aux = randi(2);    #axis (x,y)
255  switch(poinSorted)
256    case 1
257      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
258        (1-80)
259        if (poinSorted_aux == 1)
260          x1+=5;
261        else
262          y1+=5;
263        endif
264      else
265        if (poinSorted_aux == 1)
266          x1 = randi(85);
267        else
268          y1 = randi(85);
269        endif
270      endif
271    endswitch
272
273  for aux= (1:255)
274    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
275      solution(2)*aux) + (solution(3));      #a b c
276  endfor
277
278  #plot (0:255,new_vetor); #plot color map
279
280  for aux_i= (1:size_X)
281    for aux_j= (1:size_Y)
282      val_orig = (image01(aux_i,aux_j,i));
283      val_new = new_vetor(val_orig+1);
284      saida(aux_i,aux_j) = val_new;
285    endfor
286  endfor
287
288  poinSorted = randi(3);           #point to sort (p1,p2,
289  p3)
290  poinSorted_aux = randi(2);    #axis (x,y)
291  switch(poinSorted)
292    case 1
293      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
294        (1-80)
295        if (poinSorted_aux == 1)
296          x1+=5;
297        else
298          y1+=5;
299        endif
300      else
301        if (poinSorted_aux == 1)
302          x1 = randi(85);
303        else
304          y1 = randi(85);
305        endif
306      endif
307    endswitch
308
309  for aux= (1:255)
310    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
311      solution(2)*aux) + (solution(3));      #a b c
312  endfor
313
314  #plot (0:255,new_vetor); #plot color map
315
316  for aux_i= (1:size_X)
317    for aux_j= (1:size_Y)
318      val_orig = (image01(aux_i,aux_j,i));
319      val_new = new_vetor(val_orig+1);
320      saida(aux_i,aux_j) = val_new;
321    endfor
322  endfor
323
324  poinSorted = randi(3);           #point to sort (p1,p2,
325  p3)
326  poinSorted_aux = randi(2);    #axis (x,y)
327  switch(poinSorted)
328    case 1
329      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
330        (1-80)
331        if (poinSorted_aux == 1)
332          x1+=5;
333        else
334          y1+=5;
335        endif
336      else
337        if (poinSorted_aux == 1)
338          x1 = randi(85);
339        else
340          y1 = randi(85);
341        endif
342      endif
343    endswitch
344
345  for aux= (1:255)
346    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
347      solution(2)*aux) + (solution(3));      #a b c
348  endfor
349
350  #plot (0:255,new_vetor); #plot color map
351
352  for aux_i= (1:size_X)
353    for aux_j= (1:size_Y)
354      val_orig = (image01(aux_i,aux_j,i));
355      val_new = new_vetor(val_orig+1);
356      saida(aux_i,aux_j) = val_new;
357    endfor
358  endfor
359
360  poinSorted = randi(3);           #point to sort (p1,p2,
361  p3)
362  poinSorted_aux = randi(2);    #axis (x,y)
363  switch(poinSorted)
364    case 1
365      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
366        (1-80)
367        if (poinSorted_aux == 1)
368          x1+=5;
369        else
370          y1+=5;
371        endif
372      else
373        if (poinSorted_aux == 1)
374          x1 = randi(85);
375        else
376          y1 = randi(85);
377        endif
378      endif
379    endswitch
380
381  for aux= (1:255)
382    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
383      solution(2)*aux) + (solution(3));      #a b c
384  endfor
385
386  #plot (0:255,new_vetor); #plot color map
387
388  for aux_i= (1:size_X)
389    for aux_j= (1:size_Y)
390      val_orig = (image01(aux_i,aux_j,i));
391      val_new = new_vetor(val_orig+1);
392      saida(aux_i,aux_j) = val_new;
393    endfor
394  endfor
395
396  poinSorted = randi(3);           #point to sort (p1,p2,
397  p3)
398  poinSorted_aux = randi(2);    #axis (x,y)
399  switch(poinSorted)
400    case 1
401      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
402        (1-80)
403        if (poinSorted_aux == 1)
404          x1+=5;
405        else
406          y1+=5;
407        endif
408      else
409        if (poinSorted_aux == 1)
410          x1 = randi(85);
411        else
412          y1 = randi(85);
413        endif
414      endif
415    endswitch
416
417  for aux= (1:255)
418    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
419      solution(2)*aux) + (solution(3));      #a b c
420  endfor
421
422  #plot (0:255,new_vetor); #plot color map
423
424  for aux_i= (1:size_X)
425    for aux_j= (1:size_Y)
426      val_orig = (image01(aux_i,aux_j,i));
427      val_new = new_vetor(val_orig+1);
428      saida(aux_i,aux_j) = val_new;
429    endfor
430  endfor
431
432  poinSorted = randi(3);           #point to sort (p1,p2,
433  p3)
434  poinSorted_aux = randi(2);    #axis (x,y)
435  switch(poinSorted)
436    case 1
437      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
438        (1-80)
439        if (poinSorted_aux == 1)
440          x1+=5;
441        else
442          y1+=5;
443        endif
444      else
445        if (poinSorted_aux == 1)
446          x1 = randi(85);
447        else
448          y1 = randi(85);
449        endif
450      endif
451    endswitch
452
453  for aux= (1:255)
454    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
455      solution(2)*aux) + (solution(3));      #a b c
456  endfor
457
458  #plot (0:255,new_vetor); #plot color map
459
460  for aux_i= (1:size_X)
461    for aux_j= (1:size_Y)
462      val_orig = (image01(aux_i,aux_j,i));
463      val_new = new_vetor(val_orig+1);
464      saida(aux_i,aux_j) = val_new;
465    endfor
466  endfor
467
468  poinSorted = randi(3);           #point to sort (p1,p2,
469  p3)
470  poinSorted_aux = randi(2);    #axis (x,y)
471  switch(poinSorted)
472    case 1
473      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
474        (1-80)
475        if (poinSorted_aux == 1)
476          x1+=5;
477        else
478          y1+=5;
479        endif
480      else
481        if (poinSorted_aux == 1)
482          x1 = randi(85);
483        else
484          y1 = randi(85);
485        endif
486      endif
487    endswitch
488
489  for aux= (1:255)
490    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
491      solution(2)*aux) + (solution(3));      #a b c
492  endfor
493
494  #plot (0:255,new_vetor); #plot color map
495
496  for aux_i= (1:size_X)
497    for aux_j= (1:size_Y)
498      val_orig = (image01(aux_i,aux_j,i));
499      val_new = new_vetor(val_orig+1);
500      saida(aux_i,aux_j) = val_new;
501    endfor
502  endfor
503
504  poinSorted = randi(3);           #point to sort (p1,p2,
505  p3)
506  poinSorted_aux = randi(2);    #axis (x,y)
507  switch(poinSorted)
508    case 1
509      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
510        (1-80)
511        if (poinSorted_aux == 1)
512          x1+=5;
513        else
514          y1+=5;
515        endif
516      else
517        if (poinSorted_aux == 1)
518          x1 = randi(85);
519        else
520          y1 = randi(85);
521        endif
522      endif
523    endswitch
524
525  for aux= (1:255)
526    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
527      solution(2)*aux) + (solution(3));      #a b c
528  endfor
529
530  #plot (0:255,new_vetor); #plot color map
531
532  for aux_i= (1:size_X)
533    for aux_j= (1:size_Y)
534      val_orig = (image01(aux_i,aux_j,i));
535      val_new = new_vetor(val_orig+1);
536      saida(aux_i,aux_j) = val_new;
537    endfor
538  endfor
539
540  poinSorted = randi(3);           #point to sort (p1,p2,
541  p3)
542  poinSorted_aux = randi(2);    #axis (x,y)
543  switch(poinSorted)
544    case 1
545      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
546        (1-80)
547        if (poinSorted_aux == 1)
548          x1+=5;
549        else
550          y1+=5;
551        endif
552      else
553        if (poinSorted_aux == 1)
554          x1 = randi(85);
555        else
556          y1 = randi(85);
557        endif
558      endif
559    endswitch
560
561  for aux= (1:255)
562    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
563      solution(2)*aux) + (solution(3));      #a b c
564  endfor
565
566  #plot (0:255,new_vetor); #plot color map
567
568  for aux_i= (1:size_X)
569    for aux_j= (1:size_Y)
570      val_orig = (image01(aux_i,aux_j,i));
571      val_new = new_vetor(val_orig+1);
572      saida(aux_i,aux_j) = val_new;
573    endfor
574  endfor
575
576  poinSorted = randi(3);           #point to sort (p1,p2,
577  p3)
578  poinSorted_aux = randi(2);    #axis (x,y)
579  switch(poinSorted)
580    case 1
581      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
582        (1-80)
583        if (poinSorted_aux == 1)
584          x1+=5;
585        else
586          y1+=5;
587        endif
588      else
589        if (poinSorted_aux == 1)
590          x1 = randi(85);
591        else
592          y1 = randi(85);
593        endif
594      endif
595    endswitch
596
597  for aux= (1:255)
598    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
599      solution(2)*aux) + (solution(3));      #a b c
600  endfor
601
602  #plot (0:255,new_vetor); #plot color map
603
604  for aux_i= (1:size_X)
605    for aux_j= (1:size_Y)
606      val_orig = (image01(aux_i,aux_j,i));
607      val_new = new_vetor(val_orig+1);
608      saida(aux_i,aux_j) = val_new;
609    endfor
610  endfor
611
612  poinSorted = randi(3);           #point to sort (p1,p2,
613  p3)
614  poinSorted_aux = randi(2);    #axis (x,y)
615  switch(poinSorted)
616    case 1
617      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
618        (1-80)
619        if (poinSorted_aux == 1)
620          x1+=5;
621        else
622          y1+=5;
623        endif
624      else
625        if (poinSorted_aux == 1)
626          x1 = randi(85);
627        else
628          y1 = randi(85);
629        endif
630      endif
631    endswitch
632
633  for aux= (1:255)
634    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
635      solution(2)*aux) + (solution(3));      #a b c
636  endfor
637
638  #plot (0:255,new_vetor); #plot color map
639
640  for aux_i= (1:size_X)
641    for aux_j= (1:size_Y)
642      val_orig = (image01(aux_i,aux_j,i));
643      val_new = new_vetor(val_orig+1);
644      saida(aux_i,aux_j) = val_new;
645    endfor
646  endfor
647
648  poinSorted = randi(3);           #point to sort (p1,p2,
649  p3)
650  poinSorted_aux = randi(2);    #axis (x,y)
651  switch(poinSorted)
652    case 1
653      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
654        (1-80)
655        if (poinSorted_aux == 1)
656          x1+=5;
657        else
658          y1+=5;
659        endif
660      else
661        if (poinSorted_aux == 1)
662          x1 = randi(85);
663        else
664          y1 = randi(85);
665        endif
666      endif
667    endswitch
668
669  for aux= (1:255)
670    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
671      solution(2)*aux) + (solution(3));      #a b c
672  endfor
673
674  #plot (0:255,new_vetor); #plot color map
675
676  for aux_i= (1:size_X)
677    for aux_j= (1:size_Y)
678      val_orig = (image01(aux_i,aux_j,i));
679      val_new = new_vetor(val_orig+1);
680      saida(aux_i,aux_j) = val_new;
681    endfor
682  endfor
683
684  poinSorted = randi(3);           #point to sort (p1,p2,
685  p3)
686  poinSorted_aux = randi(2);    #axis (x,y)
687  switch(poinSorted)
688    case 1
689      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
690        (1-80)
691        if (poinSorted_aux == 1)
692          x1+=5;
693        else
694          y1+=5;
695        endif
696      else
697        if (poinSorted_aux == 1)
698          x1 = randi(85);
699        else
700          y1 = randi(85);
701        endif
702      endif
703    endswitch
704
705  for aux= (1:255)
706    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
707      solution(2)*aux) + (solution(3));      #a b c
708  endfor
709
710  #plot (0:255,new_vetor); #plot color map
711
712  for aux_i= (1:size_X)
713    for aux_j= (1:size_Y)
714      val_orig = (image01(aux_i,aux_j,i));
715      val_new = new_vetor(val_orig+1);
716      saida(aux_i,aux_j) = val_new;
717    endfor
718  endfor
719
720  poinSorted = randi(3);           #point to sort (p1,p2,
721  p3)
722  poinSorted_aux = randi(2);    #axis (x,y)
723  switch(poinSorted)
724    case 1
725      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
726        (1-80)
727        if (poinSorted_aux == 1)
728          x1+=5;
729        else
730          y1+=5;
731        endif
732      else
733        if (poinSorted_aux == 1)
734          x1 = randi(85);
735        else
736          y1 = randi(85);
737        endif
738      endif
739    endswitch
740
741  for aux= (1:255)
742    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
743      solution(2)*aux) + (solution(3));      #a b c
744  endfor
745
746  #plot (0:255,new_vetor); #plot color map
747
748  for aux_i= (1:size_X)
749    for aux_j= (1:size_Y)
750      val_orig = (image01(aux_i,aux_j,i));
751      val_new = new_vetor(val_orig+1);
752      saida(aux_i,aux_j) = val_new;
753    endfor
754  endfor
755
756  poinSorted = randi(3);           #point to sort (p1,p2,
757  p3)
758  poinSorted_aux = randi(2);    #axis (x,y)
759  switch(poinSorted)
760    case 1
761      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
762        (1-80)
763        if (poinSorted_aux == 1)
764          x1+=5;
765        else
766          y1+=5;
767        endif
768      else
769        if (poinSorted_aux == 1)
770          x1 = randi(85);
771        else
772          y1 = randi(85);
773        endif
774      endif
775    endswitch
776
777  for aux= (1:255)
778    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
779      solution(2)*aux) + (solution(3));      #a b c
780  endfor
781
782  #plot (0:255,new_vetor); #plot color map
783
784  for aux_i= (1:size_X)
785    for aux_j= (1:size_Y)
786      val_orig = (image01(aux_i,aux_j,i));
787      val_new = new_vetor(val_orig+1);
788      saida(aux_i,aux_j) = val_new;
789    endfor
790  endfor
791
792  poinSorted = randi(3);           #point to sort (p1,p2,
793  p3)
794  poinSorted_aux = randi(2);    #axis (x,y)
795  switch(poinSorted)
796    case 1
797      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
798        (1-80)
799        if (poinSorted_aux == 1)
800          x1+=5;
801        else
802          y1+=5;
803        endif
804      else
805        if (poinSorted_aux == 1)
806          x1 = randi(85);
807        else
808          y1 = randi(85);
809        endif
810      endif
811    endswitch
812
813  for aux= (1:255)
814    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
815      solution(2)*aux) + (solution(3));      #a b c
816  endfor
817
818  #plot (0:255,new_vetor); #plot color map
819
820  for aux_i= (1:size_X)
821    for aux_j= (1:size_Y)
822      val_orig = (image01(aux_i,aux_j,i));
823      val_new = new_vetor(val_orig+1);
824      saida(aux_i,aux_j) = val_new;
825    endfor
826  endfor
827
828  poinSorted = randi(3);           #point to sort (p1,p2,
829  p3)
830  poinSorted_aux = randi(2);    #axis (x,y)
831  switch(poinSorted)
832    case 1
833      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
834        (1-80)
835        if (poinSorted_aux == 1)
836          x1+=5;
837        else
838          y1+=5;
839        endif
840      else
841        if (poinSorted_aux == 1)
842          x1 = randi(85);
843        else
844          y1 = randi(85);
845        endif
846      endif
847    endswitch
848
849  for aux= (1:255)
850    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
851      solution(2)*aux) + (solution(3));      #a b c
852  endfor
853
854  #plot (0:255,new_vetor); #plot color map
855
856  for aux_i= (1:size_X)
857    for aux_j= (1:size_Y)
858      val_orig = (image01(aux_i,aux_j,i));
859      val_new = new_vetor(val_orig+1);
860      saida(aux_i,aux_j) = val_new;
861    endfor
862  endfor
863
864  poinSorted = randi(3);           #point to sort (p1,p2,
865  p3)
866  poinSorted_aux = randi(2);    #axis (x,y)
867  switch(poinSorted)
868    case 1
869      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
870        (1-80)
871        if (poinSorted_aux == 1)
872          x1+=5;
873        else
874          y1+=5;
875        endif
876      else
877        if (poinSorted_aux == 1)
878          x1 = randi(85);
879        else
880          y1 = randi(85);
881        endif
882      endif
883    endswitch
884
885  for aux= (1:255)
886    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
887      solution(2)*aux) + (solution(3));      #a b c
888  endfor
889
890  #plot (0:255,new_vetor); #plot color map
891
892  for aux_i= (1:size_X)
893    for aux_j= (1:size_Y)
894      val_orig = (image01(aux_i,aux_j,i));
895      val_new = new_vetor(val_orig+1);
896      saida(aux_i,aux_j) = val_new;
897    endfor
898  endfor
899
900  poinSorted = randi(3);           #point to sort (p1,p2,
901  p3)
902  poinSorted_aux = randi(2);    #axis (x,y)
903  switch(poinSorted)
904    case 1
905      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
906        (1-80)
907        if (poinSorted_aux == 1)
908          x1+=5;
909        else
910          y1+=5;
911        endif
912      else
913        if (poinSorted_aux == 1)
914          x1 = randi(85);
915        else
916          y1 = randi(85);
917        endif
918      endif
919    endswitch
920
921  for aux= (1:255)
922    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
923      solution(2)*aux) + (solution(3));      #a b c
924  endfor
925
926  #plot (0:255,new_vetor); #plot color map
927
928  for aux_i= (1:size_X)
929    for aux_j= (1:size_Y)
930      val_orig = (image01(aux_i,aux_j,i));
931      val_new = new_vetor(val_orig+1);
932      saida(aux_i,aux_j) = val_new;
933    endfor
934  endfor
935
936  poinSorted = randi(3);           #point to sort (p1,p2,
937  p3)
938  poinSorted_aux = randi(2);    #axis (x,y)
939  switch(poinSorted)
940    case 1
941      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
942        (1-80)
943        if (poinSorted_aux == 1)
944          x1+=5;
945        else
946          y1+=5;
947        endif
948      else
949        if (poinSorted_aux == 1)
950          x1 = randi(85);
951        else
952          y1 = randi(85);
953        endif
954      endif
955    endswitch
956
957  for aux= (1:255)
958    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
959      solution(2)*aux) + (solution(3));      #a b c
960  endfor
961
962  #plot (0:255,new_vetor); #plot color map
963
964  for aux_i= (1:size_X)
965    for aux_j= (1:size_Y)
966      val_orig = (image01(aux_i,aux_j,i));
967      val_new = new_vetor(val_orig+1);
968      saida(aux_i,aux_j) = val_new;
969    endfor
970  endfor
971
972  poinSorted = randi(3);           #point to sort (p1,p2,
973  p3)
974  poinSorted_aux = randi(2);    #axis (x,y)
975  switch(poinSorted)
976    case 1
977      if ((x1 <= 80) || (y1 <= 80))  #p1 interval
978        (1-80)
979        if (poinSorted_aux == 1)
980          x1+=5;
981        else
982          y1+=5;
983        endif
984      else
985        if (poinSorted_aux == 1)
986          x1 = randi(85);
987        else
988          y1 = randi(85);
989        endif
990      endif
991    endswitch
992
993  for aux= (1:255)
994    new_vetor(aux+1) = (solution(1)*(aux^2)) + (
995      solution(2)*aux) + (solution(3));      #a b c
996  endfor
997
998  #plot (0:255,new_vetor); #plot color map
999
1000 for aux_i= (1:size_X)
1001   for aux_j= (1:size_Y)
1002     val_orig = (image01(aux_i,aux_j,i));
1003     val_new = new_vetor(val_orig+1);
1004     saida(aux_i,aux_j) = val_new;
1005   endfor
1006 endfor
1007
1008 poinSorted = randi(3);           #point to sort (p1,p2,
1009 p3)
1010 poinSorted_aux = randi(2);    #axis (x,y)
1011 switch(poinSorted)
1012   case 1
1013     if ((x1 <= 80) || (y1 <= 80))  #p1 interval
1014       (1-80)
1015       if (poinSorted_aux == 1)
1016         x1+=5;
1017       else
1018         y1+=5;
1019       endif
1020     else
1021       if (poinSorted_aux == 1)
1022         x1 = randi(85);
1023       else
1024         y1 = randi(85);
1025       endif
1026     endif
1027   endswitch
1028
1029 for aux= (1:255)
1030   new_vetor(aux+1) = (solution(1)*(aux^2)) + (
1031     solution(2)*aux) + (solution(3));      #a b c
1032 endfor
1033
1034 #plot (0:255,new_vetor); #plot color map
1035
1036 for aux_i= (1:size_X)
1037   for aux_j= (1:size_Y)
1038     val_orig = (image01(aux_i,aux_j,i));
1039     val_new = new_vetor(val_orig+1);
1040     saida(aux_i,aux_j) = val_new;
1041   endfor
1042 endfor
1043
1044 poinSorted = randi(3);           #point to sort (p1,p2,
1045 p3)
1046 poinSorted_aux = randi(2);    #axis (x,y)
1047 switch(poinSorted)
1048   case 1
1049     if ((x1 <= 80) || (y1 <= 80))  #p1 interval
1050       (1-80)
1051       if (poinSorted_aux == 1)
1052         x1+=5;
1053       else
1054         y1+=5;
1055       endif
1056     else
1057       if (poinSorted_aux == 1)
1058         x1 = randi(85);
1059       else
1060         y1 =
```

```
49         endif
50
51     case 2
52         if ((x2 <= 165) || (y2 <= 165)) #p2 interval
53             (1-165)
54             if (poinSorted_aux == 1)
55                 x2+=5;
56             else
57                 y2+=5;
58             endif
59             else
60                 if (poinSorted_aux == 1)
61                     x2 = randi([85,170]);
62                 else
63                     y2 = randi([85,170]);
64                 endif
65             endif
66     case 3
67         if ((x3 <= 251) || (y3 <= 251)) #p3 interval
68             (1-251)
69             if (poinSorted_aux == 1)
70                 x3+=5;
71             else
72                 y3+=5;
73             endif
74             else
75                 if (poinSorted_aux == 1)
76                     x3 = randi([170,256]);
77                 else
78                     y3 = randi([170,256]);
79                 endif
80             endif
81             otherwise
82                 printf("Error 0x01\n")
83         endswitch
84
85         new_histogram = imhist(saida);      #out iteration
86         histogram
87         new_dif = D(new_histogram,image02_histogram);      #
88             diff between new histogram and image suport
89             histogram
90     endwhile
91
92     printf("New Channel %d...", i);
93
94     #save each channel in variables
```

```
90     switch(i)
91     case 1
92         image01_out_R = saida;
93         diff_red = new_dif;
94     case 2
95         image01_out_G = saida;
96         diff_green = new_dif;
97     case 3
98         image01_out_B = saida;
99         diff_blue = new_dif;
100    otherwise
101        printf("Error 0x02\n");
102    endswitch
103
104    saida = zeros(size_X,size_Y); #reset var saida
105    i++;                         #incremente channel
106 endwhile
107
108 image_final = cat(3,image01_out_R,image01_out_G,
109                     image01_out_B);
110 figure('name','Final Image Iteration','numbertitle',
111         'off'), imshow(uint8(image_final));
112 if ((diff_red < best_difR) && (diff_green < best_difG)
113     && (diff_blue < best_difB))
114     best_difR = diff_red;
115     best_difG = diff_green;
116     best_difB = diff_blue;
117     best_image = image_final;
118 endif
119 temp--; #decrease temperature
120 endwhile
```

Resultados

Inicialmente foram utilizadas duas imagens como entrada, sendo elas: "*data/img1_patch.png*" e "*data/img2_patch.png*", a segunda imagem foi escolhida como base de comparação de saída ou seja, a figura gerada deverá se aproximar mais da segunda imagem , tendo como ponto inicial de partida a figura "*img1_patch.png*". A seguir a figura 1 apresenta as imagens usadas como entrada:

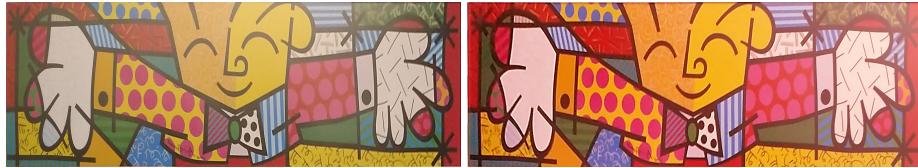


Figure 1: Conjunto das imagens originais, img1_path à direita e img2_path à esquerda.

Após carregar ambas as imagens foi necessário calcular o histograma de cada imagem separando em canais vermelho, verde e azul (RGB), sendo assim o histograma gerado contendo os três canais é apresentado a seguir pela figura 2.

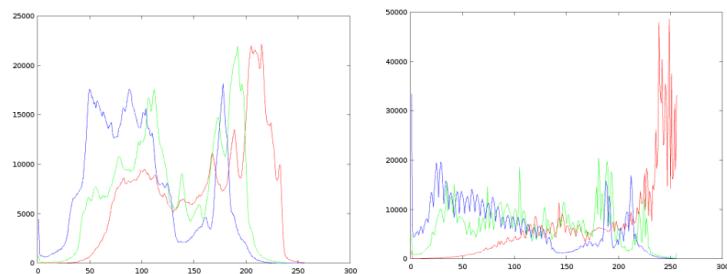


Figure 2: Histogramas referentes as imagens img1_path e img2_path respectivamente.

Para a primeira execução, foi escolhido como temperatura inicial o valor 5, após 8 a 10 minutos obtivemos os seguintes resultados como apresentado pelas imagens 3, 4, 5 e 6, não foram colocadas todas as imagens geradas a cada iteração pois foram geradas muitas e apenas as mais relevantes foram selecionadas. A imagem 3 apresentada a seguir é referente à imagem inicial antes da execução do algoritmo.



Figure 3: Imagem inicial usada como entrada do algoritmo.

A imagem apresentada pela figura 4, apresenta a imagem após o primeiro decremento da temperatura. Em comparação à imagem original de entrada é possível perceber uma leve melhora na distribuição das cores, tornando assim a imagem mais nítida.



Figure 4: Imagem gerada após a execução do arquivo "*Code.m*" com temperatura igual a 4.

A imagem seguinte apresentada pela figura 5, é referente ao segundo decremento da temperatura, sendo assim é possível notar que as cores da imagem tornaram -se mais escuras, dando a impressão que a imagem foi gerada em um ambiente com baixa luminosidade, podendo esta descrição ser observada a seguir:



Figure 5: Imagem gerada após a execução do arquivo "*Code.m*" com temperatura igual a 2.

Como resultado final, dentre todas as imagens geradas com temperatura inicial igual a cinco, o algoritmo selecionou a imagem que é apresentada pela figura 6, como sendo a imagem mais próxima da "*img2_patch.png*" utilizada como comparação.

Após a execução é possível perceber que as cores da imagem gerada em comparação à imagem inicial tornaram -se mais nítidas, aumentando assim o contraste entre as cores.



Figure 6: Resultado da melhor imagem gerada para temperatura iniciada em 5.

Para outro teste realizamos a execução do código com a variável "*temp*" iniciada em 15, na qual esta variável representa o número de iterações que o código irá realizar para encontrar a melhor faixa de cores em cada canal vermelho, verde e azul.

Sendo assim, após a execução, os resultados obtidos levaram em média de 20 à 25 minutos. A seguir serão apresentadas algumas das imagens referentes aos resultados obtidos para temperatura igual à 15. Para a terceira iteração do código com temperatura igual à 12, a imagem resultante em comparação à imagem original apresentou um escurecimento na tonalidade das cores, sendo observada pela figura 7.



Figure 7: Imagem gerada após a execução do arquivo "*Code.m*" com temperatura igual a 12.

Com temperatura igual a 9, a imagem resultante apresentou uma pequena melhora nas cores da imagem, tornando ela um pouco mais clara, sendo assim, tais observações são apresentadas pela figura 8



Figure 8: Imagem gerada após a execução do arquivo "*Code.m*" com temperatura igual a 9.

Após a conclusão do algoritmo a imagem final gerada é apresentada pela figura 9. É possível perceber que houve uma grande saturação das cores, aumentando muito sua intensidade, além de perder completamente a cor laranja no rosto do personagem principal retratado.



Figure 9: Imagem final gerada com temperatura inicial igual a 15.