

Repte 6

Alumne: Martí Caixal Joaniquet
NIU: 1563587, data: 23/04/2022

1 Introducció

Al repte 6 es demana fer una correspondència entre imatges. Hi ha 3 apartats principals:

- Detector de Harris per trobar els punts interessants
- Generar descripció per cada punt
- Fer correspondència entre els punts

Si bé només es mostraran les parts de codi més importants a cada apartat, al final del document es pot trobar el codi sencer.

2 Imatges i preprocessament

A continuació es mostren les imatges amb les que es realitza l'activitat.

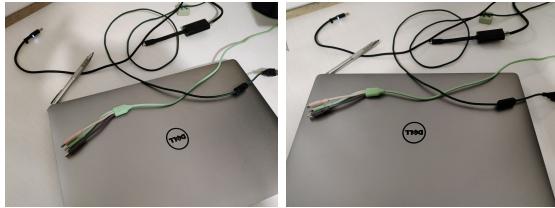


Figura 1: Imatges inicials

Per poder tenir uns temps d'execució acceptable s'ha hagut de reduir la mida de les imatges per $\frac{1}{3}$ de la resolució original.

3 Detector de Harris

La primera part del programa és crear l'algorisme de Harris, que permet trobar els punts d'interès de cada imatge. Es comença per convolucionar la imatge amb el següent filtre:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

També es convoluciona de nou fent un 'flip' al filtre. Els dos resultats s'anomenen I_x i I_y .

```
1 Ix = conv2(I(cmin:cmax, rmin:rmax), dx, 'same');  
2 Iy = conv2(I(cmin:cmax, rmin:rmax), dy, 'same');
```

A continuació es crea una gaussiana per utilitzar-la com a Gaussian Blur. La manera d'aplicar-la és, de nou, amb l'ús de convolucions utilitzant els valors anterioris:

```
1 g = fspecial('gaussian', max(1, fix(6*  
    sigma)), sigma);  
2 Ix2 = conv2(Ix.^2, g, 'same');  
3 Iy2 = conv2(Iy.^2, g, 'same');  
4 Ixy = conv2(Ix.*Iy, g, 'same');
```

Seguidament s'aplica la fórmula que permet obtenir el valor R de cada punt de la imatge. És el que indica la intensitat de cada píxel segons la seva singularitat. Un valor elevat indica que el punt és més probable a ser una cantonada.

```
1 R = (Ix2.*Iy2 - Ixy.^2) - k*(Ix2 + Iy2  
    ).^2;
```

Mitjançant un threshold es seleccionen els punts amb més intensitat.

4 Filtrar punts

El resultat aconseguit fins al moment és el següent per ambdues imatges:

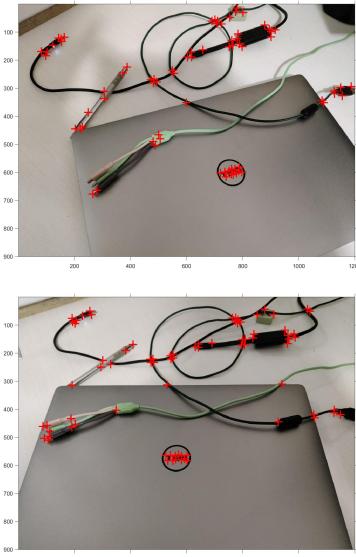


Figura 2: Tots els punts trobats

Si bé ha trobat correctament els punts que pertanyen a cantonades, hi ha bastants que estan molt propers entre ells. Es pot veure clarament al logo de la marca del portàtil. Tot i no ser un problema que impedeixi el bon funcionament del programa, es pot mirar de tractar.

Per solucionar-ho es defineix una distància mínima a la que han d'estar els punts. Com que prèviament s'ha obtingut el valor R de cada punt, s'ordenen tenint-ho en compte. Així s'aconsegueix donar preferència als que tenen un valor més gran i s'eliminen els seus veïns amb un valor més petit.

La segona correcció que es fa és eliminar els punts que estan molt propers als laterals. Un dels passos següents és obtenir una subimatge del voltant del punt. Per tant, els

punts que estan tocant un lateral no permetrien obtenir una subimatge de tots els costats.

La següent figura mostra el resultat un cop aplicades les dues correccions:

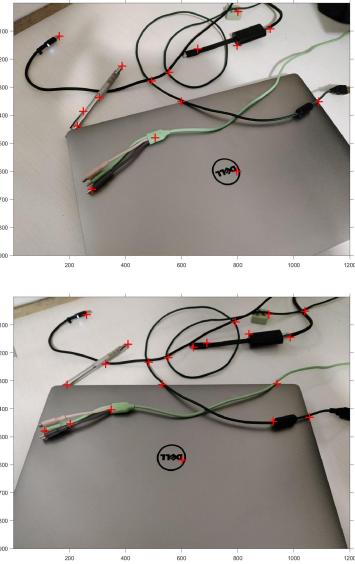


Figura 3: Punts filtrats

5 Correspondència de punts

Una vegada es té els punts de cada imatge, cal fer una correspondència entre ells. Per cada punt de la primera imatge es crea un descriptor que consisteix en la subimatge al voltant del punt en qüestió. Cada un d'aquests descriptors de la imatge 1 es compara amb tots els descriptors de la imatge 2. El més similar serà el punt de la imatge 2 corresponent al punt de la imatge 1.

S'ha programat dos mètodes per comparar els descriptors.

5.1 Correspondència per diferència del sumatori

La idea principal d'aquest mètode és sumar tots els valors del punt de la subimatge 1. Aquesta suma resultant es resta, per totes les subimatges de la imatge 2, als seus respectius sumatoris. El resultat amb el valor més petit significa que els dos punts són els més semblants d'entre tots.

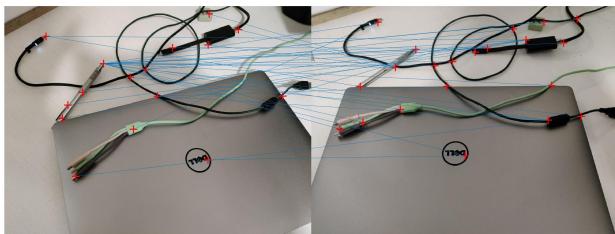


Figura 4: Resultat per diferència de sumatori

L'avantatge que té aquest mètode és que, tot i estar les subimatges amb diferents rotacions, trobarà la correspondència correcte. El desavantatge que es veig, per altra banda, és que no pot diferenciar per colors. Tot i que una subimatge sigui totalment vermella i l'altre totalment verda, si el valor final és el mateix, donarà com a correspondència correcte.

5.2 Correspondència per Mean Squared Error

La idea principal d'aquest segon mètode és comparar ambdues subimatges utilitzant el Mean Squared Error.

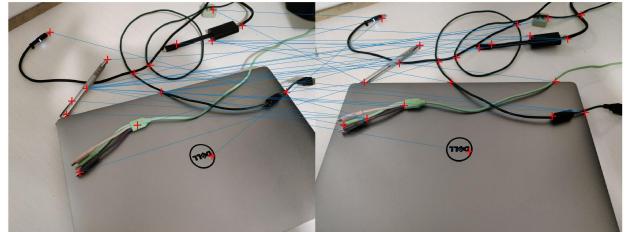


Figura 5: Resultat per Mean Squared Error

Aquest mètode es comporta al revés que l'anterior. Ara sí que permet diferenciar entre colors, però no pot fer res respecte les rotacions

Annex

```
1 clear all;
2 close all;
3 clc
4
5 search_area_px = 2;
6 n_points = 500;
7 min_distance_px = 50;
8 write = true;
9 [frame1, points1, points1_filtrats] = show_harris_points('img3.jpg', n_points,
   min_distance_px, search_area_px, write);
10 [frame2, points2, points2_filtrats] = show_harris_points('img4.jpg', n_points,
   min_distance_px, search_area_px, write);
11
12 pairs = get_matching_points_descriptor(frame1, points1_filtrats, frame2,
   points2_filtrats, search_area_px);
13 plot_lines(frame1, frame2, points1_filtrats, points2_filtrats, pairs);
14 write_image(strcat('correspondencia_desc', 'img3.jpg'), write);
15
16 pairs = get_matching_points_mse(frame1, points1_filtrats, frame2, points2_filtrats
   , search_area_px);
17 plot_lines(frame1, frame2, points1_filtrats, points2_filtrats, pairs);
18 write_image(strcat('correspondencia_mes', 'img4.jpg'), write);
19
20 function [frame, points, points_filtrats] = show_harris_points(image_name,
   n_points, min_distance_px, search_area_px, write)
% busquem tots els punts
frame1=imread(strcat('input/',image_name));
frame1 = imresize(frame1, 0.3);
frame1_points = get_harris_points(frame1, 12, n_points);
% frame1_points = sortrows(frame1_points);
figure, imshow(frame1);
axis on;
hold on;
for r=1:size(frame1_points,1)
    plot(frame1_points(r, 2), frame1_points(r, 1), 'r+', 'MarkerSize', 15,
        'LineWidth', 2)
end
write_image(strcat('all_',image_name), write);
33
34 % filtrem els que estan molt junts
35 frame1_points_filtrats = filter_similar_points(frame1_points, min_distance_px)
;
36 figure, imshow(frame1);
37 axis on;
38 hold on;
39 for r=1:size(frame1_points_filtrats, 1)
40     plot(frame1_points_filtrats(r, 2), frame1_points_filtrats(r, 1), 'r+', 'MarkerSize', 15, 'LineWidth', 2)
end
42
43 % filtrem els que estan molt a prop dels limits de la imatge
44 frame1_points_filtrats_limits = filter_edge_points(frame1_points_filtrats,
   search_area_px, size(frame1));
45 figure, imshow(frame1);
46 axis on;
47 hold on;
48 for r=1:size(frame1_points_filtrats_limits, 1)
```

```

49     plot(frame1_points_filtrats_limits(r, 2), frame1_points_filtrats_limits(r,
50           1), 'r+', 'MarkerSize', 15, 'LineWidth', 2)
51 end
52 write_image(strcat('filtered_', image_name), write);
53
54 frame = frame1;
55 points = frame1_points;
56 points_filtrats = frame1_points_filtrats_limits;
57
58 end
59
60 function plot_lines(frame1, frame2, points1, points2, pairs)
61 figure,
62 imshow([frame1, frame2]);
63 hold on
64 despl = size(frame1, 2);
65
66 for r=1:size(points1, 1)
67     plot(points1(r, 2), points1(r, 1), 'r+', 'MarkerSize', 15, 'LineWidth', 2)
68 end
69
70 for r=1:size(points2, 1)
71     plot(points2(r, 2)+despl, points2(r, 1), 'r+', 'MarkerSize', 15, 'LineWidth', 2)
72 end
73
74 for r=1:size(pairs, 1)
75     line([pairs(r, 2), pairs(r, 4)+despl], [pairs(r, 1), pairs(r, 3)]);
76 end
77
78 end
79
80 function pairs = get_matching_points_mse(frame1, frame1_points, frame2,
81 frame2_points, area)
82 pairs = zeros(1, 4);
83 index = 1;
84
85 % comparant frame1 amb frame2
86 for i=1:size(frame1_points)
87     best_mse = 99999999;
88     best_index = 0;
89     A = frame1(frame1_points(i, 1)-area:frame1_points(i, 1)+area,
90               frame1_points(i, 2)-area:frame1_points(i, 2)+area, :);
91
92     for j=1:size(frame2_points)
93         B = frame2(frame2_points(j, 1)-area:frame2_points(j, 1)+area,
94                   frame2_points(j, 2)-area:frame2_points(j, 2)+area, :);
95         mse = immse(A, B);
96         if mse < best_mse
97             best_mse = mse;
98             best_index = j;
99         end
100     end
101     pairs(index, 2) = frame1_points(i, 2);
102     pairs(index, 1) = frame1_points(i, 1);
103     pairs(index, 4) = frame2_points(best_index, 2);
104     pairs(index, 3) = frame2_points(best_index, 1);

```

```

104     index = index+1;
105
106
107 % comparant frame2 amb frame1
108 for i=1:size(frame2_points)
109     best_mse = 99999999;
110     best_index = 0;
111     A = frame2(frame2_points(i, 1)-area:frame2_points(i, 1)+area,
112                 frame2_points(i, 2)-area:frame2_points(i, 2)+area, :);
113
114     for j=1:size(frame1_points)
115         B = frame2(frame1_points(j, 1)-area:frame1_points(j, 1)+area,
116                     frame1_points(j, 2)-area:frame1_points(j, 2)+area, :);
117         mse = immse(A, B);
118         if mse < best_mse
119             best_mse = mse;
120             best_index = j;
121         end
122     end
123     pairs(index, 4) = frame2_points(i, 2);
124     pairs(index, 3) = frame2_points(i, 1);
125     pairs(index, 2) = frame1_points(best_index, 2);
126     pairs(index, 1) = frame1_points(best_index, 1);
127
128     index = index+1;
129
130 % eliminent punts duplicitats
131 pairs = unique(pairs, 'rows');
132
133 function pairs = get_matching_points_descriptor(frame1, frame1_points, frame2,
134 frame2_points, area_px)
135     pairs = zeros(1, 5);
136     index = 1;
137
138 % comparant frame1 amb frame2
139 for i=1:size(frame1_points)
140     best_result = 99999999;
141     best_index = 0;
142
143     A = frame1(frame1_points(i, 1)-area_px:frame1_points(i, 1)+area_px,
144                 frame1_points(i, 2)-area_px:frame1_points(i, 2)+area_px, :);
145     A_vector = reshape(A, 1, []);
146
147     for j=1:size(frame2_points)
148         B = frame2(frame2_points(j, 1)-area_px:frame2_points(j, 1)+area_px,
149                     frame2_points(j, 2)-area_px:frame2_points(j, 2)+area_px, :);
150         B_vector = reshape(B, 1, []);
151         res = sum(abs(B_vector - A_vector));
152         if res < best_result
153             best_result = res;
154             best_index = j;
155         end
156     end
157     pairs(index, 2) = frame1_points(i, 2);
158     pairs(index, 1) = frame1_points(i, 1);
159     pairs(index, 4) = frame2_points(best_index, 2);
160     pairs(index, 3) = frame2_points(best_index, 1);
161     pairs(index, 5) = best_result;

```

```

159     index = index+1;
160 end
161
162 % comparant frame2 amb frame1
163 for i=1:size(frame2_points)
164     best_result = 99999999;
165     best_index = 0;
166
167     A = frame2(frame2_points(i, 1)-area_px:frame2_points(i, 1)+area_px,
168                 frame2_points(i, 2)-area_px:frame2_points(i, 2)+area_px, :);
169     A_vector = reshape(A, 1, []);
170
171     for j=1:size(frame1_points)
172         B = frame1(frame1_points(j, 1)-area_px:frame1_points(j, 1)+area_px,
173                     frame1_points(j, 2)-area_px:frame1_points(j, 2)+area_px, :);
174         B_vector = reshape(B, 1, []);
175         res = sum(abs(B_vector - A_vector));
176         if res < best_result
177             best_result = res;
178             best_index = j;
179         end
180     end
181     pairs(index, 4) = frame2_points(i, 2);
182     pairs(index, 3) = frame2_points(i, 1);
183     pairs(index, 2) = frame1_points(best_index, 2);
184     pairs(index, 1) = frame1_points(best_index, 1);
185
186     index = index+1;
187 end
188
189 % eliminent punts duplicitats
190 pairs = unique(pairs, 'rows');
191
192 end
193
194 function new_points = filter_similar_points(frame_points, max_distance_px)
195     new_points = zeros(1, 2);
196     index = 1;
197     for i=1:size(frame_points,1)-1
198         separat = 1;
199         a = [frame_points(i, 2) frame_points(i, 1)];
200         for j=i+1:size(frame_points,1)
201             b = [frame_points(j, 2) frame_points(j, 1)];
202             euclidean_distance = norm(b-a);
203             if euclidean_distance < max_distance_px
204                 separat = 0;
205                 break;
206             end
207         end
208         if separat == 1
209             new_points(index, 2) = frame_points(i, 2);
210             new_points(index, 1) = frame_points(i, 1);
211             index = index + 1;
212         end
213     end
214
215 end
216 new_points = unique(new_points, 'rows');

```

```

217 end
218
219 function new_points = filter_edge_points(frame_points, min_distance_px,
220 frame_dimensions)
220 new_points = zeros(1, 2);
221 index = 1;
222
223 for i=1:size(frame_points, 1)
224 if frame_points(i, 1)-min_distance_px < 1 || frame_points(i, 1) +
224 min_distance_px > frame_dimensions(1)
225 continue;
226 end
227 if frame_points(i, 2)-min_distance_px < 1 || frame_points(i, 2) +
227 min_distance_px > frame_dimensions(2)
228 continue;
229 end
230 new_points(index, 2) = frame_points(i, 2);
231 new_points(index, 1) = frame_points(i, 1);
232 index = index + 1;
233 end
234 new_points = unique(new_points, 'rows');
235 end
236
237 function points = get_harris_points(frame, min_N, max_N)
238 I =double(frame);
239 ymin = 7;
240 ymax = size(frame, 2)-7;
241 xmin = 7;
242 xmax = size(frame, 1)-7;
243 Aj=6;
244 cmin=xmin-Aj; cmax=xmax+Aj; rmin=ymin-Aj; rmax=ymax+Aj;
245
246 sigma=2; Thrshold=20; r=6; disp=1;
247 dx = [-1 0 1; -1 0 1; -1 0 1];
248 dy = dx';
249
250 Ix = conv2(I(cmin:cmax, rmin:rmax), dx, 'same');
251 Iy = conv2(I(cmin:cmax, rmin:rmax), dy, 'same');
252
253 g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);
254
255 Ix2 = conv2(Ix.^2, g, 'same');
256 Iy2 = conv2(Iy.^2, g, 'same');
257 Ixy = conv2(Ix.*Iy, g, 'same');
258 k = 0.04;
259 R11 = (Ix2.*Iy2 - Ixy.^2) - k*(Ix2 + Iy2).^2;
260 R11=(1000/max(max(R11)))*R11;
261 R=R11;
262 ma=max(max(R));
263 sze = 2*r+1;
264 MX = ordfilt2(R, sze^2, ones(sze));
265 R11 = (R==MX)&(R>Thrshold);
266 count=sum(sum(R11(5:size(R11,1)-5,5:size(R11,2)-5)));
267
268 loop=0;
269 while ((count<min_N) || (count>max_N))&&(loop<30)
270 if count>max_N
271 Thrshold=Thrshold *1.5;
272 elseif count < min_N
273 Thrshold=Thrshold *0.5;

```

```

274 end
275
276 R11 = (R==MX)&(R>Thrshold);
277 count=sum(sum(R11(5:size(R11,1)-5,5:size(R11,2)-5)));
278 loop=loop+1;
279
280 R=R*0;
281 R(5:size(R11,1)-5,5:size(R11,2)-5)=R11(5:size(R11,1)-5,5:size(R11,2)-5);
282 [r1,c1] = find(R);
283 PIP=[r1+cmin, c1+rmin];
284
285 points = PIP;
286
287
288 end
289
290 function write_image(name, save)
291 if save == true
292     saveas(gcf, strcat('./output/', name));
293 end
294 end

```