**Automatizar la creación de un mosaico de fotos**

APELLIDOS, NOMBRE: MUÑOZ NAVARRO, ALEJANDRO

**LAB 1: (Obtención de keypoints en imágenes usando el algoritmo SIFT)**

Haced un volcado con el nº de "keypoints" encontrados en el resto de las imágenes.

164 237 223 175 79 117 28 252

Mostrar la 5ª imagen superponiendo sobre ella los "keypoints" encontrados en ella como círculos rojos ('ro'). Adjuntar imagen.

Imagen que contiene interior, diferente

Descripción generada automáticamente

**LAB 2: (Detectar posibles emparejamientos entre pares de imágenes)**

Volcad el nº de parejas encontradas entre la segunda imagen y todas las demás.

Adjuntar vuestro código de find\_matches.m

imagen 1: puntos 41

imagen 3: puntos 38

imagen 4: puntos 26

imagen 5: puntos 57

imagen 6: puntos 32

imagen 7: puntos 12

imagen 8: puntos 21

imagen 9: puntos 6

imagen 10: puntos 0

imagen 11: puntos 0

imagen 12: puntos 0

function [xy1, xy2] = find\_matches(s1, s2)

N1 = size(s1.xy,1);

N2 = size(s2.xy,1);

parejas = zeros(N2,1);

for k = 1:N1

M = 0;

d1 = s1.id(k,:);

for j= 1:N2

if(parejas(j)>0)

else

d2=s2.id(j,:);

m = 1./norm(d1-d2);

if(m > M)

J = j;

M = m;

end

end

end

if(M > 4)

parejas(J) = k;

end

end

index2 = find(parejas > 0);

index1 = parejas(index2);

xy1 = s1.xy(index1,:);

xy2 = s2.xy(index2,:);

end

**LAB 3: (Errores en el ajuste)**

A partir de (xy1) cread una matriz de coord. homogéneas (1ª fila = x's, 2ª fila=y's, 3ª fila=1's). Aplicad P a esta matriz (multiplicando) y volcad las coordenadas u,v obtenidas. ¿Por qué no son iguales que las coordenadas deseadas (xy2)?

u = 86 318 599 200 451

v = 445 69 375 184 154

Son casi iguales, tan sólo tiene un error de ajuste.

Adjuntar código de vuestra función error\_ajuste().

function err=error\_ajuste(P,xy1,xy2)

H = [xy1';xy1(:,1)'.^0];

uv = P\*H;

u=uv(1,:)./uv(3,:);

v=uv(2,:)./uv(3,:);

du=abs(u-xy2(:,1)');

dv=abs(v-xy2(:,2)');

err = sqrt(du.^2+dv.^2)';

end

Dad el valor del error para los 3 otros puntos del ejemplo anterior.

0.4390 0.5264 1.2317

**LAB 4: (RANSAC)**

Adjuntad código de vuestra función ransac().

function [T Nok]=ransac(xy1,xy2)

NP = size(xy2,1);

ok = [];

for k = 1:1000

exit=0;

while(exit==0)

idx = floor((NP-1).\*rand(4,1) + 1);

idx=unique(idx,'stable');

if(length(idx)==4)

exit=1;

end

end

xy3 = xy1(idx,:);

xy4 = xy2(idx,:);

P=get\_proy(xy3,xy4);

error=error\_ajuste(P,xy1,xy2);

cerca=find(error<0.5);

if(length(cerca) > length(ok))

ok = cerca;

T = P;

end

end

Nok = length(ok);

end

Hallar el ajuste (con get\_proy) usando todos los datos y volcad la matriz T obtenida. Hacer un plot del error cometido para cada uno de los puntos (la salida de error\_ajuste). Adjuntad la gráfica obtenida.

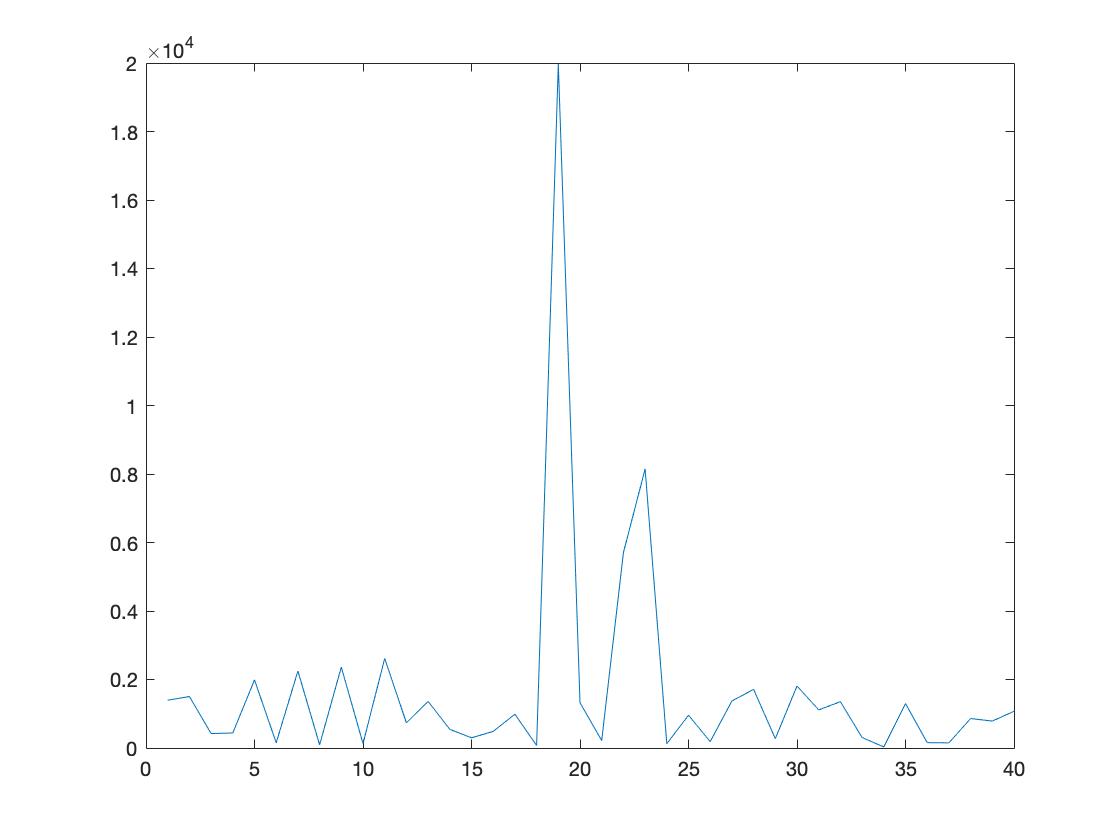
T =

1.0e+03 \*

-0.0086 -0.0098 5.7094

-0.0106 -0.0093 6.1236

0.0000 -0.0000 0.0010



Repetir ahora el ajuste pero ahora usando la función ransac(). ¿Cuántos puntos han sido detectados como "inliers" por ransac? Volcad la nueva matriz T de ajuste, que debe ser mucho más parecida a la ideal. Adjuntad el nuevo gráfico de los errores usando la nueva matriz de transformación. Interpretarlo.

Nok = 20

T =

0.6998 0.3009 99.9868

-0.2002 0.7507 199.9951

0.0010 -0.0200 1.0000

Imagen que contiene texto

Descripción generada automáticamente

El error aparece en los últimos 20 puntos.

**LAB 5: (Determinar la conectividad entre las imágenes).**

Adjuntar código de find\_QP() y volcar la matriz Q de "conectividad" obtenida.

function[Q, P]=find\_QP(s)

NF = size(s,2);

Q=zeros(NF);

P=cell(NF);

M = 0;

R = 0;

for i=1:NF

for j=i+1:NF

[xy1, xy2] = find\_matches(s{i},s{j});

if(size(xy1,1)>10)

M = M + size(xy1,1);

[T, Nok]=ransac(xy1,xy2);

R = R + Nok;

if(Nok>=8)

Q(i,j) = Nok;

Q(j,i) = Nok;

P{i,j}=T;

P{j,i}=inv(T);

end

end

end

end

end

Q =

0 38 0 0 24 96 24 17 0 0 0 0

38 0 35 24 53 31 11 18 0 0 0 0

0 35 0 31 22 0 0 0 12 0 0 0

0 24 31 0 50 0 0 31 44 24 14 0

24 53 22 50 0 26 29 66 33 34 11 12

96 31 0 0 26 0 69 29 0 0 0 23

24 11 0 0 29 69 0 51 0 32 0 88

17 18 0 31 66 29 51 0 43 58 15 27

0 0 12 44 33 0 0 43 0 33 15 0

0 0 0 24 34 0 32 58 33 0 15 39

0 0 0 14 11 0 0 15 15 15 0 0

0 0 0 0 12 23 88 27 0 39 0 0

Volcad la matriz de transformación P entre coordenadas de la imagen 1 --> 5 y la que pasa de coordenadas de la imagen 7 a la 1.

1 🡪 5

1.1986 -0.1530 -408.2630

0.1929 0.9802 320.0596

0.0003 -0.0003 1.0000

7 🡪 1

0.4352 -0.0109 468.2333

-0.0440 0.6738 -33.4705

-0.0003 0.0000 0.6906

¿Qué porcentaje de parejas ha aceptado RANSAC sobre las detectadas iniciamente?

91.3220

**LAB 6:**

Adjuntar el código usado para detectar la imagen ancla.

clear;

clc;

load keypoints;

load QP\_data;

% Buscamos la fila que mas se repite sin valores nulos

[row,col] = find(Q);

ancla = row(1);

contador = sum(row == ancla);

puntos = sum(Q(ancla,:));

for k = 2:size(row,1)

if(ancla~=k)

if(sum(row == k)>contador)

contador = sum(row == k);

ancla = k;

puntos = sum(Q(k,:));

% Si coincide con otra en repeticiones comparamos el n˙mero de puntos

elseif(sum(row == k)==contador && sum(Q(k,:))>puntos)

puntos = sum(Q(k,:));

ancla = k;

end

end

end

¿Qué imagen habéis escogido como ancla? ¿Con cuántas imágenes está conectada?

He escogido el índice 5, es decir, la imagen número 5. Está conectada a 11 imágenes con 359 parejas confirmadas.

Adjuntar el código de enlaza() y los resultados para las matrices T{1},T{2},T{3}.

function T=enlaza(Q,P)

NF = size(Q,1);

% Buscamos la fila que mas se repite sin valores nulos

[row,col] = find(Q);

ancla = row(1);

contador = sum(row == ancla);

puntos = sum(Q(ancla,:));

for k = 2:size(row,1)

if(ancla~=k)

if(sum(row == k)>contador)

contador = sum(row == k);

ancla = k;

puntos = sum(Q(k,:));

% Si coincide con otra en repeticiones comparamos el n˙mero de puntos

elseif(sum(row == k)==contador && sum(Q(k,:))>puntos)

puntos = sum(Q(k,:));

ancla = k;

end

end

end

T = cell(1,NF);

proc = false(1,NF);

T{ancla}=eye(3);

proc(ancla)=true;

while (size(find(not(proc))) ~=0)

usadas = find(proc);

sin\_usar = find(not(proc));

A = Q(sin\_usar,usadas);

[M, pos]=max(A(:));

pos=pos(1);

[i,j]=ind2sub(size(A),pos);

T{sin\_usar(i)} = T{usadas(j)} \* P{sin\_usar(i),usadas(j)};

proc(sin\_usar(i))=true;

end

end

T{1} =

1.1831 -0.1451 -403.4543

0.1744 0.9684 327.6974

0.0002 -0.0003 0.9958

T{2} =

1.2721 -0.1039 -446.2205

0.1892 1.1847 -124.9214

0.0003 -0.0000 1.0000

T{3} =

1.3169 -0.0415 -442.0100

0.1630 1.2604 -488.5707

0.0003 0.0002 0.9267

**LAB 7:**

Dar los valores finales obtenidos para u\_min, u\_max, v\_min, v\_max

u\_min = -597 u\_max = 2325 v\_min = -988 v\_max = 1387

Dar los desplazamiento dU, dV a usar así como el ANCHO/ALTO del mosaico.

dU = 598 dV = 989 ANCHO = 2923 ALTO = 2376

Adjuntad matriz de transformación T{1} una vez incorporada la translación (dU,dV).

He aplicado un format longG para visualizar mejor el resultado:

1.32872859623716 -0.322054196608046 192.015863612726

0.415261809991588 0.675708594988982 1312.51344420025

0.000243500367134572 -0.000295973512337486 0.995769521378355

Adjuntad mosaico resultado (imshow).



Adjuntad el mosaico final (eliminando zonas negras entre las fotos)

Imagen que contiene naturaleza

Descripción generada automáticamente

Recortar el mosaico eliminando las zonas negras de alrededor. Adjuntad resultado.

Imagen que contiene exterior, cielo, naturaleza

Descripción generada automáticamente

Además de ir incluyendo el código pedido en el fichero de respuestas, adjuntar un **directorio con TODO el código necesario (SIN incluir las imágenes) para que funcione vuestro programa. Entregadlo en un .7z o similar junto con esta hoja de respuestas (en formato pdf).**

--------------------- Hasta aquí, 70% de la nota --------------------

**Fusión de exposiciones (30%)**

**Para esta parte he decidido hacer diferentes soluciones con un switch que corresponde a cada parte del ejercicio y poder comparar mejor los resultados.**

Adjuntad código usado para la fusión de las pirámides.

NF = 3;

name='gala\_';

x = input("Escribe una opción:\n(1) Básica.\n(2) Ajuste.\n(3) Normalización por canal y saturación.\n--> ");

switch x

case 1

P=cell(1,NF);

NL = 6;

for k = 1:NF

fich = sprintf('%s%d.jpg',name,k);

im = imread(fich);

P{k} = lap(im,NL);

end

A = P{1};

B = P{2};

C = P{3};

medias=P{1};

medias{NL} = (A{NL}+B{NL}+C{NL})./3;

for k=1:NL-1

im = medias{k};

for i = 1:size(im,1)

for j = 1:size(im,2)

im(i,j) = max([A{k}(i,j) B{k}(i,j) C{k}(i,j)]);

end

end

medias{k}=im;

end

im = invlap(medias);

Adjuntad el código usado para re-escalar la imagen y aplicarle la ecualización adaptativa (y opcionalmente el resalte de la saturación).

% Primero ajustamos el valor por el mÌnimo si es necesario

if(min(im)<0)

im = im-min(im);

end

% DespuÈs ajustamos el valor por el m·ximo si es necesario

if(max(im)>1)

im = im./max(im);

end

% Cambiamos a HSV y aplicamos funciÛn

fprintf('E. Retoques de Brillo/Contraste/Saturacion\n');

im=rgb2hsv(im);

clip = 0.01;

im(:,:,3) = adapthisteq(im(:,:,3),'ClipLimit',clip);

p=0.7;

im(:,:,2)=im(:,:,2).^p;

im=hsv2rgb(im);

figure(1);

hold on, title("Caso 1"), imshow(uint8(im\*255)), hold off;

Adjuntad imagen final resultante, acompañada por un zoom de la zona de la cresta.

Imagen que contiene valle, montaña, cañón, naturaleza

Descripción generada automáticamenteImagen que contiene foto, naturaleza

Descripción generada automáticamente

**Registro previo de imágenes**

Indicad el nº puntos encontrados en cada imagen.

S{1} = 306 S{2} = 220 S{3} = 443

Indicad el nº de emparejamientos encontrados y volcad las matrices T2 y T3.

Número de emparejamientos = 186

T{2} =

1.00272961201416 -0.0047871260109449 -1.30109704514287

0.00303519269818683 1.00121735752392 -1.7387401909407

1.53661169328418e-06 -1.54223613941589e-06 1.00000066563315

T{3} =

1.00424700809012 -0.00622410967293022 -3.08920257682343

0.00131911355202467 1.00348986384889 -3.03855066428291

3.5488767598109e-06 -4.46530098117768e-06 1.00000251934421

Adjuntad la nueva imagen resultante.

Imagen que contiene montaña, valle, naturaleza, cañón

Descripción generada automáticamente

Haced un zoom en la parte de la cresta y adjuntarlo. Debe de haber desaparecido el problema que había de la falta de alineamiento de las imágenes.

Imagen que contiene valle, naturaleza, cañón, foto

Descripción generada automáticamente

Usar la normalización independiente por canales y experimentar con los parámetros. Adjuntad alguno de vuestros resultados que os guste, indicando los parámetros usados.

Imagen que contiene montaña, valle, naturaleza, exterior

Descripción generada automáticamente

Luminancia: 0.006

Saturación (1): 0.87

**Adjuntad dentro de vuestro .7z o similar otro directorio con vuestro código fusion.m junto con el resto de las funciones necesarias para que funcione.**