

VISIÓ PER COMPUTADOR

SHORT PROJECT

Roger Sánchez Enebral
UPC FIB Juny 2024

Index

Index	0
1. Introducció	2
1.1 Descripció de les etapes del programa	2
1.2 Problemes trobats i la seva solució	3
2. Extracció del vector de característiques	4
2.1 Selecció de punts d'interès:	4
2.2 Vector de característiques	4
2.3 Mida i temps de càcul	4
3. Resultats obtinguts en les imatges de test	5
3.1 Barack Obama	5
3.2 Tom Holland	8
3.3 Jennifer Lawrence	10
4. Funcions utilitzades	12
5. Codi	13
Learn descriptors from reference images	13
Play video with matching features	16
Extract descriptors manually from interest points	20
Manual matching of features by distance	23
6. Enllaços	24
• Carpeta a matlab drive amb tots els fitxers	24
• Video mostrant els resultats	24

1. Introducció

En aquest informe mostrarem el procés i resultats de la implementació Short Project de Visió per Computador sobre el seguiment de característiques facials en persones, tot implementant descriptors de punts singulars de proposta pròpia i el seu corresponent matching amb una seqüència de vídeo.

1.1 Descripció de les etapes del programa

El programa es divideix en 3 seccions principals:

- **Aprendre els descriptors de les imatges de referència:**

Amb imatges escollides prèviament de la persona a estudiar en diferents posicions, l'usuari escull manualment les faccions característiques que es volen trobar posteriorment en diferents imatges o vídeos. Es guarden els vectors de característiques associats a aquestes faccions, per així poder comparar-les posteriorment i fer matching amb altres noves imatges.

- **Comparar amb altres imatges:**

Una vegada apreses les característiques de les zones escollides, es carrega la imatge amb la qual es vol fer matching. Es troben els punts singulars més rellevants amb extractSURFFeatures per tal d'augmentar l'eficiència i no haver de calcular les característiques de cada pixel de la imatge, i després es calculen els vectors de característiques d'aquests punts. Finalment, es comparen tots els vectors de característiques de les imatges de referència amb els vectors de la nova imatge per la distància, s'agafa la imatge de referència amb més matches, i es seleccionen els punts que coincideixen amb dita imatge.

- **Seqüència de vídeo:**

Similar a l'anterior pas, es realitza aquesta mateixa seqüència però cada X número de frames (depenent de la velocitat a la que es mou la persona del vídeo), i es marquen constantment els punts de coincidència.

1.2 Problemes trobats i la seva solució

Al llarg de la implementació del programa ens hem trobat amb molts problemes. Els més importants a destacar són els següents:

- **El cost de fer matching a cada frame del video.** Al principi es va intentar calcular el vector de característiques per cada punt de cada frame del video i trobar els seus matches òptims. No obstant, ràpidament es va veure que aquest procés era massa costos, pel que s'ha hagut de fer optimitzacions. El resultat final ha sigut adonar-se que no fa falta fer match en cada frame del video, doncs el moviment de les persones no és tan accelerat. Per tant, fer matching cada X número de frames (dependent de la velocitat a la que es mou la persona) és més que suficient per arribar a un bon resultat. A més, es pot limitar l'àrea a la que es busca fer match sabent prèviament la zona per la que estarà la cara de la persona en qüestió, pel que no fa falta mirar punts fora d'aquesta zona. Finalment, s'ha optat per fer una elecció de punts singulars previa, i només fer matching sobre aquests punts ja seleccionats. S'ha provat amb diferents mètodes, com Harris, KAZE i SURF, i finalment s'ha escollit SURF com la opció més consistent.
- **Elecció de les característiques.** Degut a que hi ha múltiples faccions de la cara de les que s'ha de fer match, trobar característiques que serveixen per totes a la vegada i que funcionin per totes ha sigut també un repte. Finalment, amb les característiques escollides el nas de les persones no es reconeix correctament.
- **Elecció de paràmetres de threshold i neighbours.** Threshold representa la distància mínima necessària entre dos vectors de característiques per tal que es consideri match, i neighbours és el número de pixels veïns que es tenen en compte per calcular aquestes característiques. Escol·lir els paràmetres adequats per que funcionin de manera general a cada vídeo ha resultat impossible, i s'ha hagut d'ajustar aquests paràmetres per cada vídeo individualment. Això és degut a que la distància de la cara de la persona al vídeo influeix directament en la quantitat de veïns que has d'escollir per representar correctament una facció de la cara, i el threshold també depèn de la similitud de la cara donada amb les imatges de referència apreses.

2. Extracció del vector de característiques

2.1 Selecció de punts d'interès:

Com bé hem explicat abans, hem aplicat SURF per detectar ràpidament els punts d'interès. Altres mètodes han sigut provats com Harris, KAZE i SIFT, però SURF és el que millor resultats donava amb molt bona eficiència.

2.2 Vector de característiques

- L'histograma de colors HSV, que descriu la contribució dels colors en la regió donada pels veïns en la imatge. Per aconseguir això només cal fer un histograma amb cadascun dels 3 canals de HSV per separat, amb 36 capses, tenint pel color una desviació de com a màxim 10 graus.
- Local Binary Pattern, que calcula les textures dels píxels en aquesta mateixa regió. Ve amb una funció propia així que la implementació es trivial.
- Els Gradients Direccionals, que capturen la distribució dels gradients en la regió donada. Calculada també de manera trivial.

Finalment, el vector de característiques és la concatenació d'aquests tres dades normalitzades. S'ha plantejat afegir més, com la intensitat de l'escala de grisos, però aquesta es veia eclipsada per l'histograma de colors HSV.

2.3 Mida i temps de càlcul

En quant a la mida del vector de característiques, aquesta es pot calcular per punt singular de la següent manera (suposant que els veïns son $20 \times 20 = 400$):

Gradient: amb 225 veïns, G_x té 400 valors, i G_y té 400 valors també, amb una suma total de $400 + 400 = 800$ valors.

HSV: amb 36 capses per canal i 3 canals, té un total de $36 \times 3 = 108$ valors

LBP: amb tamany de la finestra, per tant, 400 valors.

Sumant els tres descriptors, el tamany total del vector de característiques és de 1308 valors.

En quant al temps de càlcul, el càlcul dels tres descriptors depèn directament del tamany del número de veïns, tal que si el número de veïns es n , el cost es $O(n^2)$ per tots tres. Si tenim en compte que hem de fer aquesta operació per cada punt singular trobat amb SURF, el cost total si el número de punts singulars són p és de $O(p^*n^2)$.

3. Resultats obtinguts en les imatges de test

3.1 Barack Obama

Aquest són els punt singulars que hem seleccionat prèviament com a candidats usant SURF:



Figura 1

I els resultats són els següents:

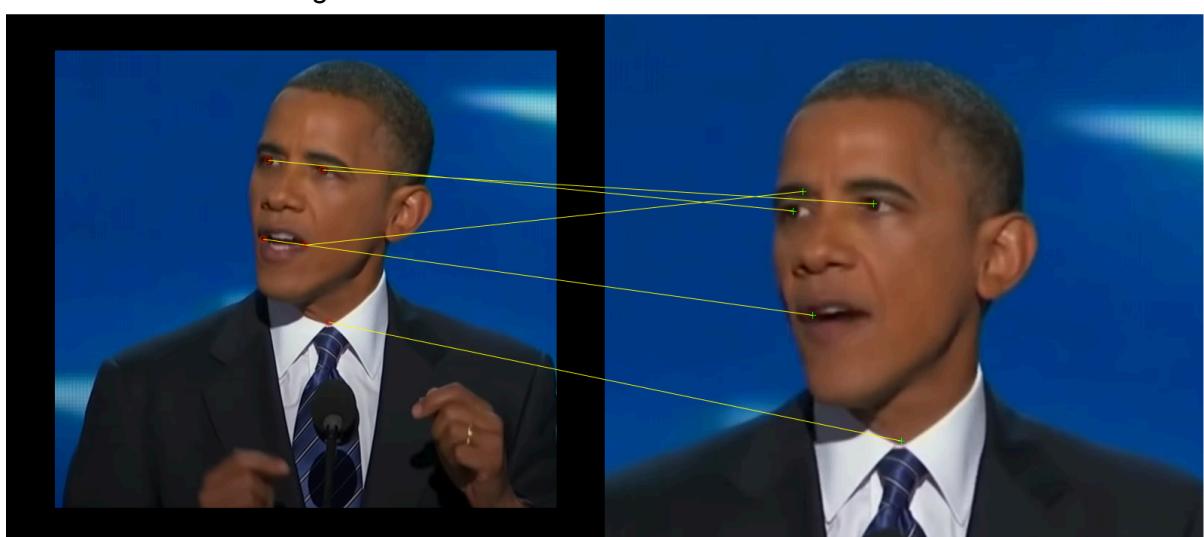


Figura 2

Podem veure com el matching s'executa correctament excepte per una comisura de la boca (Figura 2). A més a més, com a data curiosa, en totes les proves que he realitzat al llarg d'aquest projecte, la camisa i la corbata s'han seleccionat correctament totes de les vegades, probablement degut al fort contrast que hi ha entre el color de la pell i la camisa blanca, i sempre en el mateix angle.



Figuras 3 i 4

En aquest dos exemples (Figura 3 i 4) vull remarcar la importància de trobar el threshold adequat per cada cas (Distància màxima entre dos descriptors amb la qual es considera que ha sigut un match). En el primer cas el threshold es 1.5, i en el segon 1.25. Podem veure com, al tenir un threshold més gran en el primer cas, es troben més matches en una imatge de referència que no és l'òptima amb molts error de matching. No obstant, si baixem el threshold al punt ideal, es troben només els matches correctes encara que hi hagin menys.

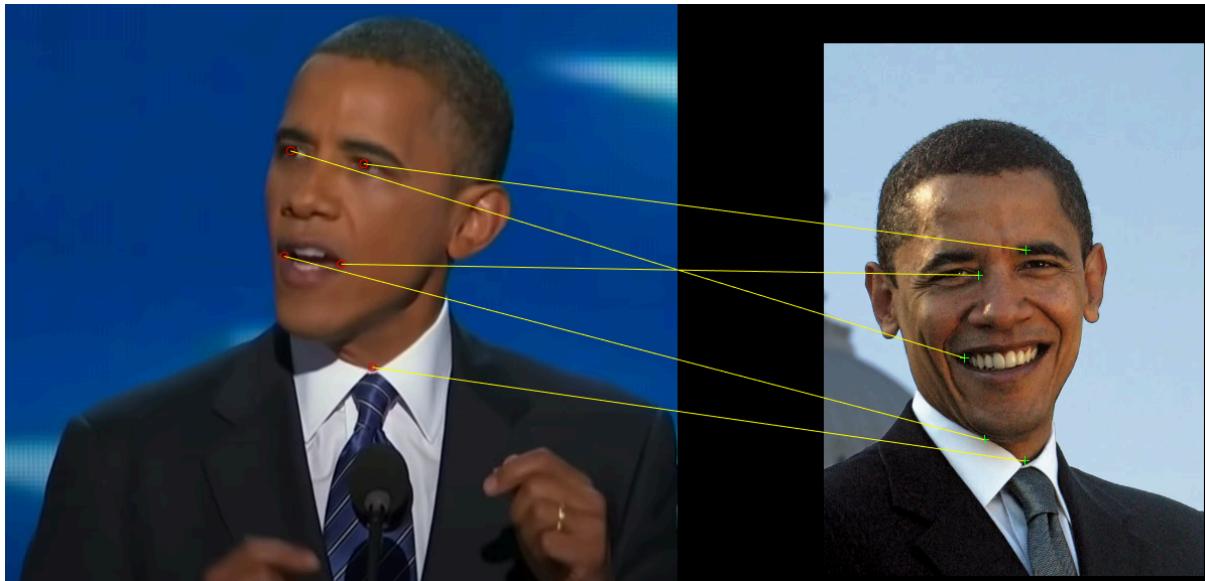


Figura 5

No obstant, a la que ens separem del set de referència que tenim, els resultats es tornen erronis (Figura 5). Per tant, un set de referència més gran, amb diferents distàncies, il·luminacions, angles i fins i tot qualitat d'imatge seria necessari per cobrir totes les possibilitats i poder fer matching en diferents entorns. Per tant, a partir d'ara ens limitarem a agafar imatges corresponents a les imatges de referència.

3.2 Tom Holland

Resultat dels punts singulars del SURF (Figura 6):



Figura 6

Es pot veure clarament el principal problema d'algunes imatges, que és la quantitat massiva de punts singulars que es donen en llocs on no es troben els punts que busquem. En aquest cas això és degut al l'estil de cabell de Tom Holland i a la brillantor que aquest reflexa. Això fa que, degut a la quantitat massiva de punts externs, molts dels matches per probabilitat es vegin cap a aquestes zones, degut a la diferència de proporció entre els punts que busquem i els que no. Això es podria millorar buscant un mètode millor per acotar els punts singulars possibles.

Tot seguit mostrem els resultats:



Figura 7

Podem veure clarament en la figura 7 que, tot i la immensa quantitat de punts exteriors, fa un matching prou bo. Tot i així no es pot evitar que alguns punts (com ara el nas, que és de les característiques de la cara que més m'ha costat trobar un bon matching), es vegin cap al cabell.



Figura 8

Per últim, si acotem el threshold correctament, podem arribar a un punt que només es mostren matchings correctes (Figura 8), però a costa de que altres features no es trobin a la imatge. A més, cada imatge, il·luminació i distància requereixen una quantitat de neighbours i un threshold diferent, pel que és molt difícil trobar uns valors que serveixin per a tots els casos.

3.3 Jennifer Lawrence

En aquest set de prova agafem els possibles punts singulars mitjançant Harris i no SURF. Com es mostra a continuació, els resultats són similars:

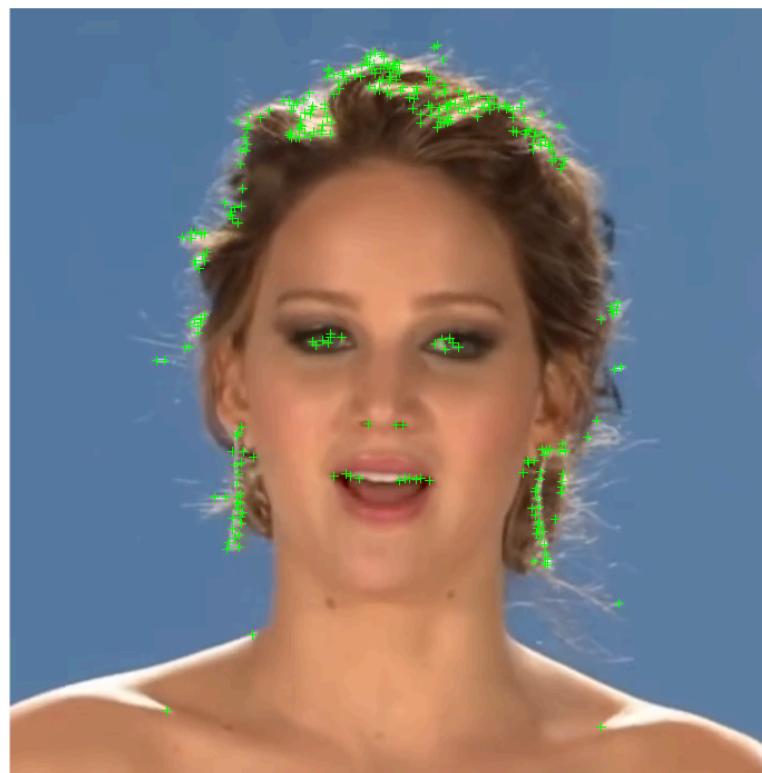


Figura 9

Podem veure a la figura 9 com els punts rellevants sofreixen una mica del mateix problema que en el cas de Tom Holland, però no tant. Per tant, s'aplica tot lo comentat en l'exemple anterior.

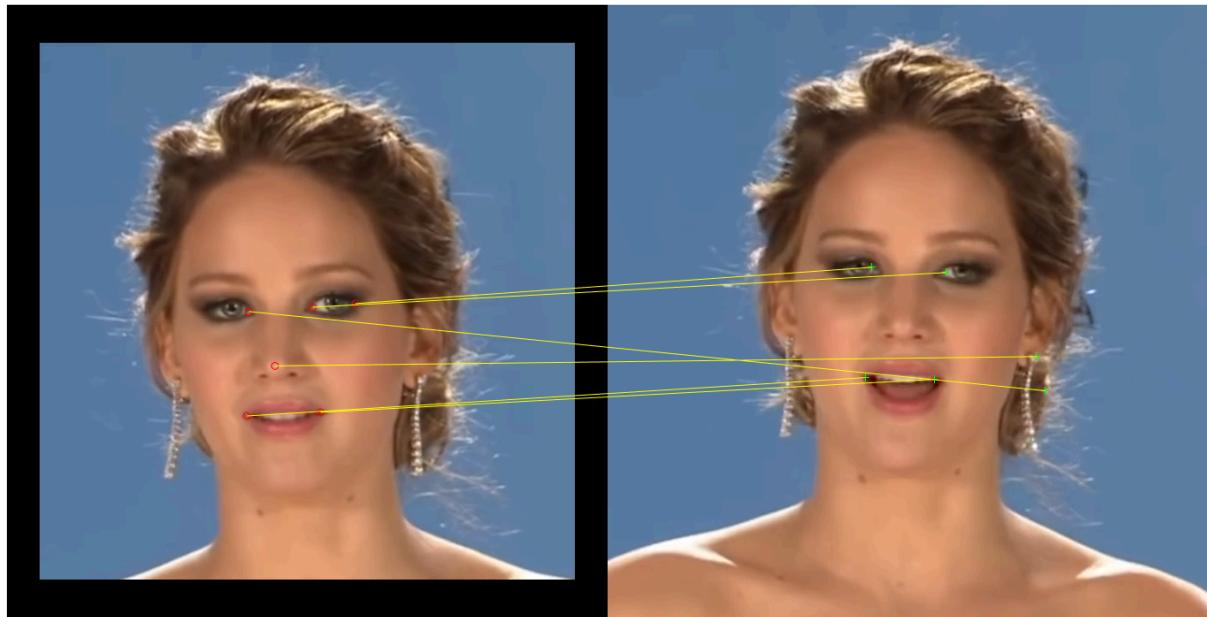


Figura 10



Figura 11

En aquests dos casos (Figura 10 i 11) es pot veure clarament com, al tenir múltiples imatges de referència, es troba la imatge més semblant a cada exemple. En la primera s'agafa com a imatge de referència una imatge de front, i a la segona es troba una imatge amb la mateixa orientació que la corresponent.

4. Funcions utilitzades

- **ExtractManualFeatures:** Implementada personalment, rep com a paràmetres una imatge, uns punts i un tamany de veïns. Aquesta funció calcula per cada punt donat de la imatge els seu vector de característiques, tenint en compte els veïns d'aquests amb tamany passat pel paràmetre, i retorna els seus descriptors. Les característiques i com calcular-les han sigut explicades prèviament en l'apartat “Extracció del vector de Característiques”.
- **ManualMatchFeatures:** Implementada també personalment, és una funció sencilla que, per cada descriptor de referència, calcula la distància amb tots els de la imatge donada, agafa el menor, i si aquest és menor a un threshold donat, l'afegeix a la llista de matches. Finalment, retorna aquesta llista.
- **detectSurfFeatures:** En quant a funcions de llibreries externes, s'ha utilitzat detectSurfFeatures per raons ja explicades, i s'ha provat també d'utilitzar per aquesta mateixa raó detectHarrisFeatures i detectKAZEFeatures.
- He mirat també múltiples problemes petits sense una referència clara sobre com interactuar amb vídeos en MATLAB i com grabar-los, però res decisiu per la realització del projecte.
- **extractLBPFeatures:** Finalment, he trobat a internet i acabat implementant extractLBPFeatures per extreure la textura dels veïns dels punts al vector de característiques, la qual cosa ha ajudat a fer un matching més correcte.

5. Codi

Enllaç del drive on es troba el codi:

<https://drive.mathworks.com/sharing/a3f7c3be-7f77-4679-a804-177f9d590b2f>

Learn descriptors from reference images

```
clear

% Reference image

referenceImagesDir = uigetdir(pwd, 'Select a folder');

referenceImages = dir(fullfile(referenceImagesDir, '*.png'));

% Storage for the reference descriptors and locations

referenceDescriptorsList = cell(length(referenceImages), 1);

referencePointsList = cell(length(referenceImages), 1);

% size of neighbour pixels that will be taken into account in the

% descriptor features

sizeOfFeatures = 20;

for i = 1:length(referenceImages)

    referenceImage = imread(strcat(referenceImagesDir, '\',
referenceImages(i).name));

    imshow(referenceImage);

    % Select manually the key points

    [x, y] = ginput;

    selectedPoints = [x, y];

    % Extract features manually

    referenceDescriptors = extractManualFeatures(referenceImage,
selectedPoints, sizeOfFeatures);

    referenceDescriptorsList{i} = referenceDescriptors;

    referencePointsList{i} = selectedPoints;

end
```

Compare different images

```
newImage = imread("imageComparisons\BarackObamaComparison.PNG");  
imshow(newImage)
```

```
% Extract possible interest points by using surf (other methods  
% have been tried and this seems the optimal one)  
surfPointsVideo = detectSURFFeatures(rgb2gray(newImage));
```

```
imshow(newImage);  
hold on;  
plot(surfPointsVideo);
```

```
surfPoints = surfPointsVideo.Location;
```

```
% we extract the descriptors from this interest points  
newImageDescriptors = extractManualFeatures(newImage, surfPoints,  
sizeOfFeatures);  
bestMatchCount = 0;  
bestReferenceImage = [];  
bestMatchedRefPoints = [];  
bestMatchednewImagePoints = [];
```

```
% For each reference we calculate its matching and if it has the  
% most matches we store it  
for i = 1:length(referenceDescriptorsList)
```

```
    referenceImage = imread(strcat(referenceImagesDir, '\\",  
    referenceImages(i).name));
```

```
    referenceDescriptors = referenceDescriptorsList{i};  
    referencePoints = referencePointsList{i};
```

```
% Manual matching

[matchedRefPoints, matchedNewImagePoints] =
manualMatchFeatures(referenceDescriptors, referencePoints,
newImageDescriptors, surfPoints, 1.3);
```

```
% Number of matches

matchCount = size(matchedRefPoints, 1);

% Update results

if matchCount > bestMatchCount

    bestMatchCount = matchCount;

    bestReferenceImage = referenceImage;

    bestMatchedRefPoints = matchedRefPoints;

    bestMatchedNewImagePoints = matchedNewImagePoints;

end
```

```
end

% Visualizar las coincidencias con la mejor imagen de referencia
encontrada

if ~isempty(bestReferenceImage)

    figure;

    showMatchedFeatures(bestReferenceImage, newImage,
bestMatchedRefPoints, bestMatchedNewImagePoints, 'montage');

end
```

Play video with matching features

```
% Read a video frame and run the face detector.

videoReader = VideoReader('PATH\NameOfInputVideo.mp4');

videoFrame = readFrame(videoReader);

videoPlayer = vision.VideoPlayer();

frameRate = videoReader.FrameRate;

resultingVideo = VideoWriter('PATH\NameOfOutputVideo.mp4', 'MPEG-4');

resultingVideo.FrameRate = videoReader.FrameRate;

open(resultingVideo);

%we calculate each x frames as it is not needed to do each one to get
a

%good result and it improves efficiency

frequencyOfMatching = 3;

frames = 0;

% We erase some of the borders of the video to improve efficiency, as
the

% face will not be there

resize = 1.5;

videoFrame = imresize(videoFrame, 1/resize);

[f, c, ~] = size(videoFrame);

borderWidth = c/4;

borderHeight = f/10;

rectLength = c-borderWidth*2;

rectHeight = f-borderHeight*4;

croppedBorder = [borderWidth borderHeight rectLength rectHeight];
```

```

% each iteration is a frame

while hasFrame(videoReader)

    frameStart = tic;

    % get the next frame

    videoFramePre = videoFrame;

    videoFrame = readFrame(videoReader);

    videoFrame = imresize(videoFrame, 1/resize);

    % Show rectangle

    %annotatedFrame = insertShape(videoFrame, 'Rectangle',
croppedBorder, 'Color', 'yellow', 'LineWidth', 10);

    %imshow(annotatedFrame)

    % each x frames we do the matching again

    if mod(frames, frequencyOfMatching) == 0

        borderlessFrame = imcrop(videoFrame, croppedBorder);

        % Extract possible interest points by using surf (other
methods

        % have been tried and this seems the optimal one)

        surfPointsVideo =
detectSURFFeatures(rgb2gray(borderlessFrame));

        surfPoints = surfPointsVideo.Location;

        % we extract the descriptors from this interest points

        videoFrameDescriptors = extractManualFeatures(borderlessFrame,
surfPoints, sizeOfFeatures);

        bestMatchCount = 0;

```

```

bestMatchedVideoFramePoints = [];

% For each reference we calculate its matching and if it has
the

% most matches we store it

for i = 1:length(referenceDescriptorsList)

    referenceDescriptors = referenceDescriptorsList{i};

    referencePoints = referencePointsList{i};

    % Manual matching

    [matchedRefPoints, matchedVideoFramePoints] =
manualMatchFeatures(referenceDescriptors, referencePoints,
videoFrameDescriptors, surfPoints, 1.25);

    % Number of matches

    matchCount = size(matchedRefPoints, 1);

    % Update results

    if matchCount > bestMatchCount

        bestMatchCount = matchCount;

        bestMatchedVideoFramePoints = matchedVideoFramePoints;

    end

end

end

% Show matches with a marker

if ~isempty(bestMatchedVideoFramePoints)

    borderlessFrame = insertMarker(borderlessFrame,
bestMatchedVideoFramePoints, 'o', 'Color', 'green');

```

```

end

videoFrame(borderHeight:borderHeight+rectHeight,
borderWidth:borderWidth+rectLength, :) = borderlessFrame;

writeVideo(resultingVideo, videoFrame);

step(videoPlayer, videoFrame);

%end of frame

frames = frames + 1;

%control of speed so it runs at most the frame rate of the video
frameDuration = toc(frameStart);
stopVideo = (1/frameRate) - frameDuration;
if stopVideo > 0
    pause(stopVideo);
end

end

% Clear videos
close(resultingVideo);
release(videoPlayer);

```

Extract descriptors manually from interest points

```
function allDescriptors = extractManualFeatures(image, points,
neighboursSize)

grayImage = rgb2gray(image);

% Start descriptor arrays
grayScaleDescriptors = [];
gradientDescriptors = [];
hsvHistogramDescriptors = [];
lbpDescriptors = [];

% neighbour size each side
halfSize = floor(neighboursSize/2);
[f, c] = size(image);

% for each point we get its descriptors
for i = 1:size(points, 1)

    x = round(points(i, 1));
    y = round(points(i, 2));

    % dont take into account pixels off the image
    if halfSize < y && f - halfSize >= y && halfSize < x && c -
halfSize >= x

        %region evaluated
        neighbours = image(y-halfSize:y+halfSize,
x-halfSize:x+halfSize, :);
        grayNeighbours = grayImage(y-halfSize:y+halfSize,
x-halfSize:x+halfSize, :);
```

```

%DESCRIPTORS:

% Color histogram HSV normalized.

hsvHistogramDescriptor = [];

neighboursHSV = rgb2hsv(neighbours);

for channel = 1:3

    histHSV = imhist(neighboursHSV(:, :, channel), 36);

    histHSV = histHSV / sum(histHSV);

    hsvHistogramDescriptor = [hsvHistogramDescriptor;
histHSV];

end

hsvHistogramDescriptor = hsvHistogramDescriptor(:)';

% gray scale normalized (not working properly as HSV seems
like

% a strict upgrade

%grayScaleDescriptor = double(grayNeighbours(:))';

%grayScaleDescriptor = grayScaleDescriptor /
norm(grayScaleDescriptor);

% Gradient normalized

[Gx, Gy] = imgradientxy(grayNeighbours);

gradientDescriptor = [Gx(:)', Gy(:)'];

gradientDescriptor = gradientDescriptor /
norm(gradientDescriptor);

% LBP (not done manually) for textures

lbpDescriptor = extractLBPFeatures(grayNeighbours);

```

```
%grayScaleDescriptors = [grayScaleDescriptors;  
grayScaleDescriptor];  
  
gradientDescriptors = [gradientDescriptors;  
gradientDescriptor];  
  
hsvHistogramDescriptors = [hsvHistogramDescriptors;  
hsvHistogramDescriptor];  
  
lbpDescriptors = [lbpDescriptors; lbpDescriptor];  
  
end  
  
end  
  
allDescriptors = [gradientDescriptors, hsvHistogramDescriptors,  
lbpDescriptors];  
  
end
```

Manual matching of features by distance

```
% Simple euclidean distance from reference descriptors to video

% descriptors

function [matchedRef, matchedVideoFrame] =
manualMatchFeatures(refDescriptors, refPoints, videoFrameDescriptors,
videoFramePoints, threshold)

matchedRef = [];

matchedVideoFrame = [];

for i = 1:size(refDescriptors, 1)

    refDescriptor = refDescriptors(i, :);

    % Euclidean distance

    distances = pdist2(videoFrameDescriptors, refDescriptor);

    [minDistance, index] = min(distances);

    % We create a threshold of distance so it has to match at
    % least for

    % a minimum distance

    if minDistance < threshold

        matchedRef = [matchedRef; refPoints(i, :)];
        matchedVideoFrame = [matchedVideoFrame;
        videoFramePoints(index, :)];
    end
end
end
```

6. Enllaços

- Carpeta a matlab drive amb tots els fitxers

<https://drive.mathworks.com/sharing/a3f7c3be-7f77-4679-a804-177f9d590b2f>

- Video mostrant els resultats

<https://youtu.be/Exqj0bQ2U1M>