

UNIVERSITAT POLITÈCNICA DE
CATALUNYA

VISIÓ PER COMPUTADOR

Short Project

Detecció elements anatòmics de la cara



Q2 2023-24, Juny 2024
Arnaud Claramunt Soler

Índex

1	Introducció	2
2	Estructura del Codi	3
3	Mètode extracció Vector Característiques	4
4	Funcions utilitzades	5
4.1	Funcions pròpies	5
4.2	Funcions ja implementades	5
5	Resultats en els Imatges Test	6
6	Resultats en els Videos Test	15
7	Casos que el programa no funciona	16
7.1	Problemes trobats i les solucions	16
8	Comparació amb mètodes existents	17
9	Annex (Codi)	18

1 Introducció

Aquest projecte té com a objectiu fer un Sistema semi-automàtic per detectar elements anatòmics de la cara utilitzant *MATLAB*. 

Els elements a cercar s'indiquen inicialment de forma manual com a key points, en una o varíes imatges de referència. D'aquesta manera es dona llibertat a l'usuari per seleccionar els punts que després es vol veure el tracking en el vídeo. Per seleccionar-los hi ha dues maneres, fer-ho sobre el primer frame del vídeo o amb una imatge de referència. En el codi està separat i ja s'indica el que s'ha de posar per utilitzar cada un.

Un exemple d'elements anatòmics podria ser: comissures dels llavis, comissures dels ulls, celles, o no anatòmica: el nus de la corbata o altres. En algun cas també he provat a la zona del nas, o la intersecció entre el nas i la línia de la galta.

Les parts principals a implementar són: un **descriptor de punts singulars (vector features)** adaptat al problema de proposta pròpia. Durant el projecte se n'ha utilitzat varis i ja s'expliquen més endavant i també els seus resultats. I una funció d'**aparellament de punts singulars (matching)**.

Junt amb l'entrega d'aquest informe, també he fet un vídeo on es veu l'execució del codi i s'explica pas per pas com funciona.

2 Estructura del Codi

L'estructura dels codis està dividida en 3 fitxers, el del codi principal, el de la funció per calcular el vector de característiques i el de la funció de calcular el matxing. Aquests codis complets es poden trobar a la secció dels Annexos.

En primer lloc comentaré les parts que conté el fitxer principal, que he anomenat *tracking.mlx*.

En el codi està ben comentat cada secció i que fa cada cosa, però explico per sobre els diferents passos que segueixo.

- Inicialització del detector de cares, cosa que utilitzo el que teníem disponible a la carpeta de l'assignatura.
- Un boolea molt important per diferenciar si volem seleccionar els punts en el primer frame del video (true) o en una imatge de referència (false).
- Llegir el primer frame del vídeo i retallar la part detectada de la cara i reescalarla a 256x256.
- Seleccionar els punts anatòmics.
- Dibuixar els punts anatòmics al frame o imatge original.
- Calcular el vector de característiques dels punts.
- Secció exclusiva si tenim imatge test, on cal fer el matxing dels punts seleccionats en una imatge test, als mateixos punts però del primer frame. És crucial fer-ho per començar el tracking del video, sinó estaria mirant uns punts d'una imatge, però volem que ho miri al video.

Part iterativa per cada frame del video:

- Llegir el frame, fer el mateix que abans, retallar i escalar la cara.
- Calcular el matxing dels punts amb el frame anterior, amb la cara actual. On a dins la funció ja calcula el vector de features una altra vegada.
- Mostrar amb creuetes els punts trackejats.

Per altre banda el fitxer *calculate_features.m* és el que implementa l'extracció de característiques i obté el vector amb els punts singulars. En parlaré en detall a la següent secció, així que ara explico com funciona el fitxer per fer el càcul del matxing(*calculate_match.m*).

El matching ha estat implementat en codi porpi, en primer lloc havia pensat en usar el *Hungarian Algorithm* per fer el matching, ja que et troba la similitud entre els vectors, però al final vaig fer una implementació molt més senzilla. Com que ja tenia la cara, els punts seleccionats i el vector de característiques, només calia buscar els punts que poden ser més similars en una finestra al voltant dels punts seleccionats.

Aquesta és una implementació especialment eficient, ja que sabem que els punts no es trobaran molt lluny. I en el meu cas, la finestra li passava com a parametre a la funció i un valor que m'ha funcionat bé és de 5. Seria un quadrat 5x5 i he probat altres valors, però aquest és el que donava resultats bons i en temps raonable.

La part important és el doble bucle *for* del final on estem mirant per cada punt la zona de la finestra i calculat per aquell punt els features. I amb la distància euclidiana, respecte els features originals, obtenir un valor. I ens guardarem els punts que fan el valor maxim. Fent que tinguem un punt en la cara que s'assembla molt al del vector de features. Aquests punts màxims que retorna la funció seran els que es dibuixaran amb creuetes vermelles en el video.

3 Mètode extracció Vector Característiques

Per fer l'extracció dels punts singulars o característiques, donats els punts seleccionats per l'usuari al fitxer de la funció `calculate_features.m` he usat descriptors no molt complexos. El que era important és que havien de ser pensats per mi i no usar alguns ja implementats. En primer lloc vaig anar a la manera fàcil i ràpida que era calcular el color RGB dels punts seleccionats, però això donava uns resultats molt dolents.

I per això vaig pensar en usar color HSV això sí, només els components H i S, perquè no depenguessin de la iluminació com vam veure en la pràctica de detecció del contorn usant l'algorisme K-means i color HSV.

Com que un sol punt pot tenir un color molt similar als dels seus veïns, vaig pensar en afegir al vector, també els adjacents, que serien els píxels superior, inferior, esquerra i dreta del seleccionat. Però no em van donar uns resultats decents pel que buscava, i vaig pensar que seria millor fer com una gran creu, o símbol de suma, just com marquem visualment els punts seleccionats per l'usuari. Ja que tenia en ment que a la part de la comissura del ull funcionaria bé, així s'estendria una mica i podria detectar la part blanca de dins o l'arruga de fora.

En resum, el primer feature són els punts seleccionats els components H i S i els seus 8 veïns adjacents en forma de suma.

La segona part del càlcul de les característiques volia calcular una cosa més robusta i com havíem vist a classe vaig pensar que seria bona idea treballar amb els histogrames dels gradients, el HOG. I utilitzant la funció ja implementada del MATLAB. Al final de tot ho afegeixo tot per columnes, i ja tenim el vector de característiques calculat.

En tot moment he intentat tenir una mida del vector no molt gran i tenint en compte el temps de càcul. Ja que si afegia més punts o característiques o coses més complexes. Tot tardaria molt més i tot i haver fet proves, em va semblar que els resultats obtinguts ja eren decents.

Altre idea que crec que funcionaria bé, és la distància entre els punts seleccionats més propers, per exemple entre els dos punts dels extrems dels ulls, mantindran una mida en porporció sempre igual, independentment de l'escala, en termes de si s'apropa o allunya a la càmera, si girés la cara seria una altra història.

4 Funcions utilitzades

Aquest apartat serà molt breu, però el volia incloure, ja que així aclareixo una part important del projecte que era usar funcions pròpies o mencionar les ja implementades, ja que sinó es consideraria plagi.

4.1 Funcions pròpies

Tot el que he explicat en les dues seccions anteriors, ha estat implementat per mi. No hi ha grans càlculs i les funcions que he definit són pel càlcul de vector de característiques amb tots els features explicats i una altra pel matxing de punts.

4.2 Funcions ja implementades

En el codi faig ús del detector de cares *Cascade Object Detector* ja vist a classe.

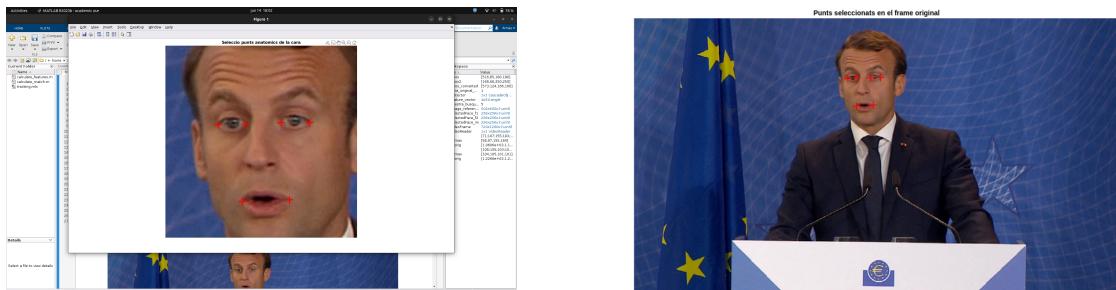
També per la segona característica he utilitzat el càlcul de HOG del MATLAB:
extractHOGFeatures.

I la resta ja són coses implementades dins el MATLAB pel seu correcte funcionament, la menys comuna pot ser la de *round* per exemple, però tot coses vistes a classe. En tot moment he volgut fer un codi senzill i no usar altres funcions.

5 Resultats en els Imatges Test

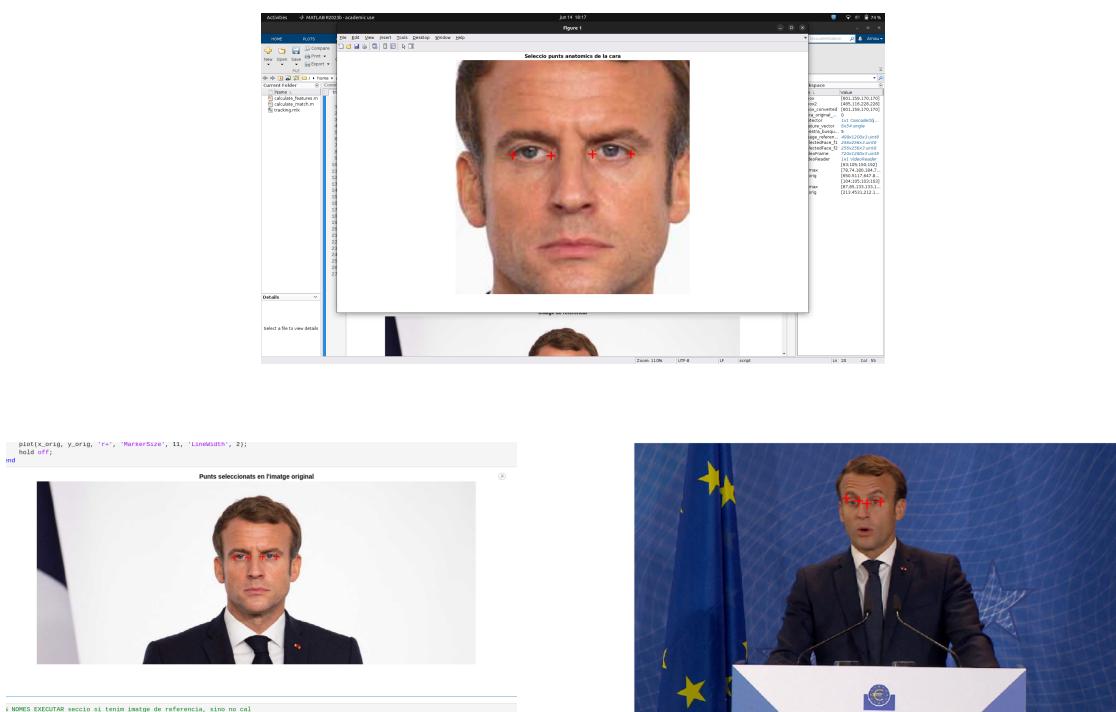
En aquesta part mostraran el procés de seleccionar alguns punts anatòmics i observar si fa un bon matching amb el primer frame. Per comprovar l'eficàcia podria calcular els Falsos positius o Falsos Negatius, si estiguéssim fent una classificació amb una xarxa neuronal per exemple. Però en el meu cas contaré el nombre de Punts que considero incorrectament classificats o simplement els mencionaré.

Començaré amb el cas del discurs del president de França el Emmanuel Macron. Que és el primer que vaig provar i també vaig pensar que al ser un polític seria més fàcil, ja que en tot moment està mirant cap a la càmera i aquesta està estàtica.



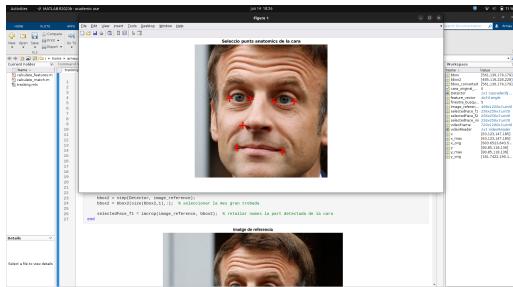
En aquest cas, l'imatge test era el frame original i no ha fet cap fallada el matxing.

També mencionar que en algunes imatges seleccionava només alguns punts anatòmics variats i així provar també el correcte funcionament



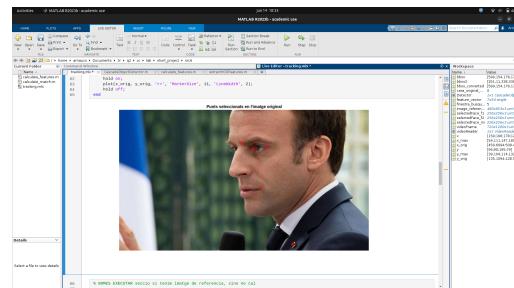
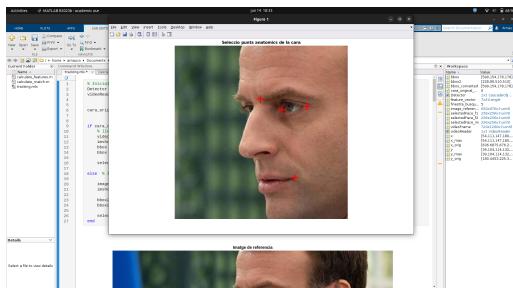
Amb aquesta imatge també funciona bé. Tot i que en la detecció al primer frame (discurs de la dreta), l'exterior de l'ull esquerre, ha anat una mica cap a la cella.

En aquest cas també he seleccionat la intersecció entre la galta i el nas com a punt.

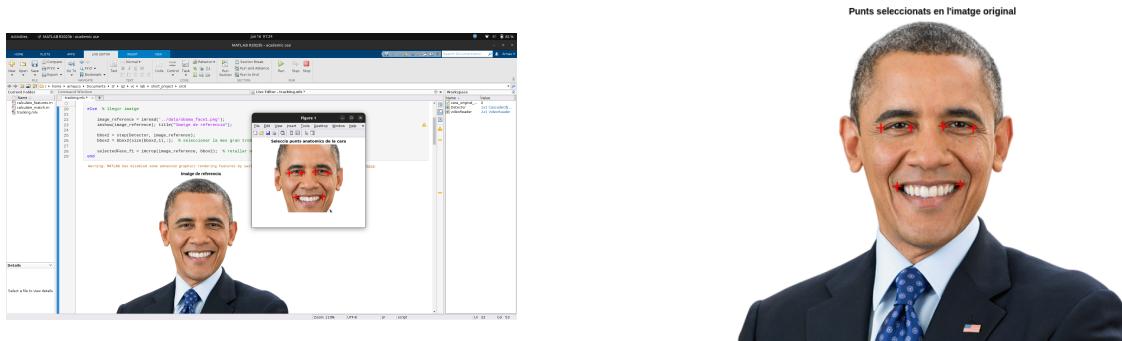


Aquí el cas incorrecte ha estat la comissura esquerra del llavi.

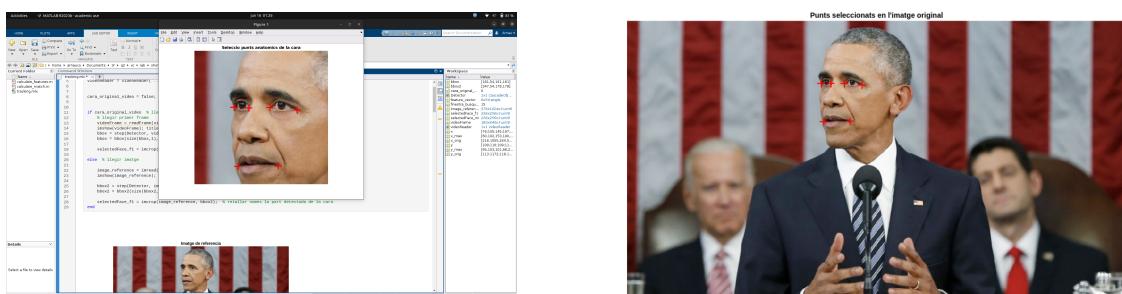
Ara una foto de perfil per posar al límit el meu detector, en aquest cas només he seleccionat coses de la part dreta de la cara, i em va sorprendre que fes un matxing tan bo.



Obama:

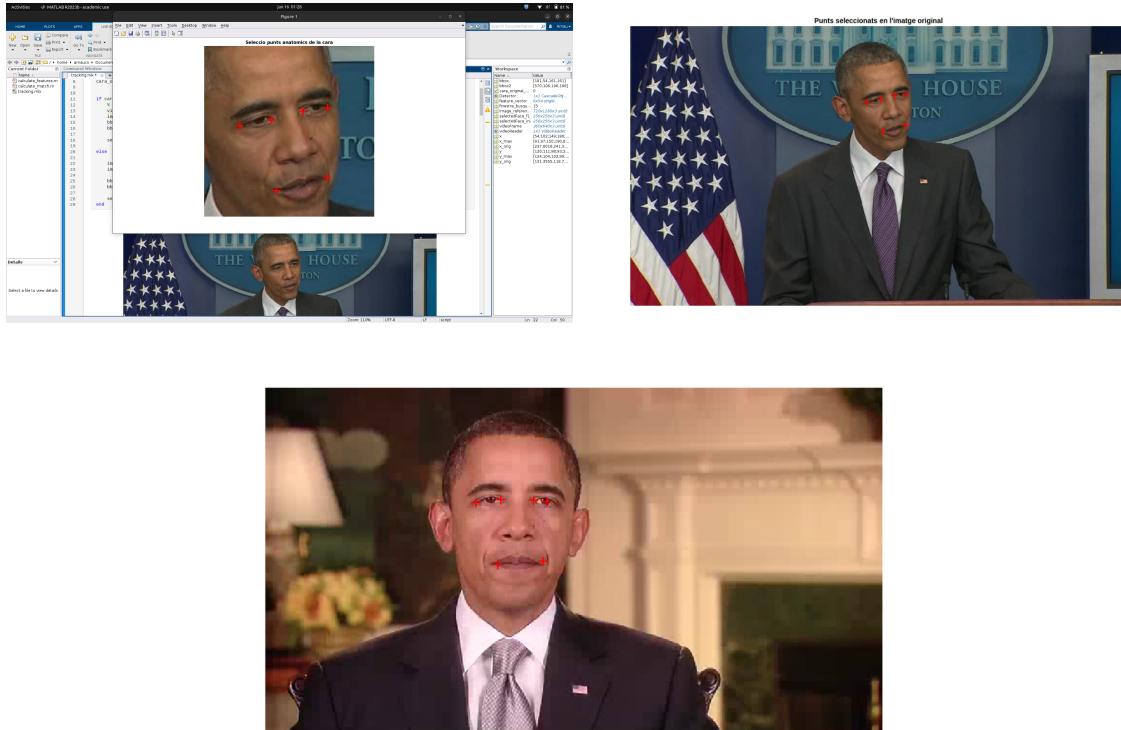


L'únic detall que li veig és que els extrems exteriors dels ulls estan més elevats del compte. Però tot i això com veurem en el següent apartat dels resultats al vídeo, l'Obama és el que m'ha donat millors resultats.



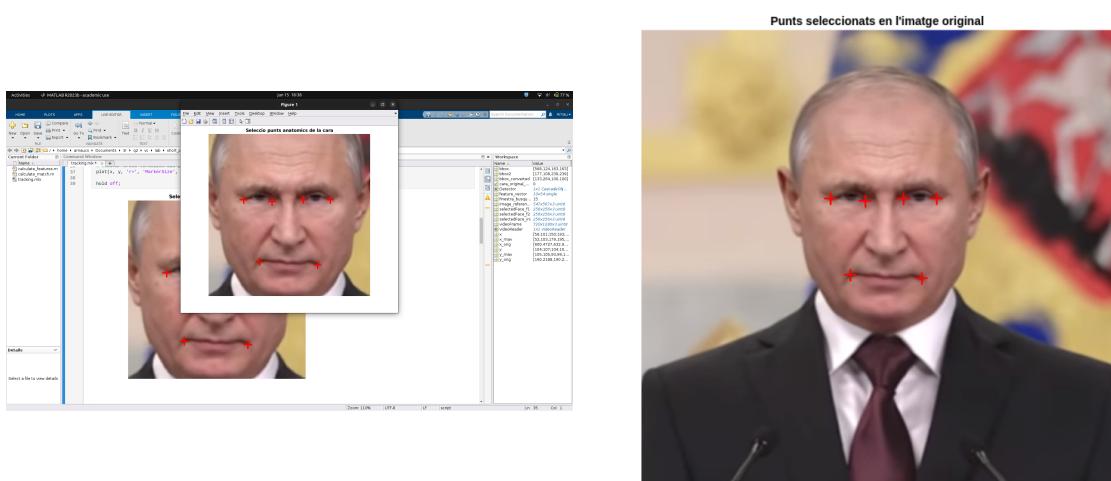
Aquí ha fallat molt la part exterior de l'ull esquerre. Però és degut a que a l'estar de perfil a la

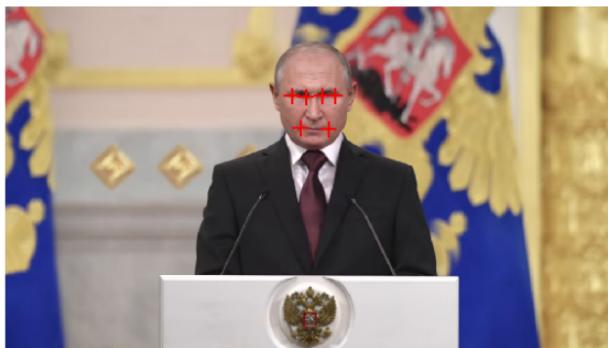
imatge de test, agafa pels veïns la part exterior de la imatge que és blanca, i per això no va bé. Però la resta tret de la part dreta del llavi ho detecta força bé. També dir que es pot repetir el procés de selecció de punts, i fer-ho amb més precisió o recalcular el matxing inicial, per veure si hi ha més sort.



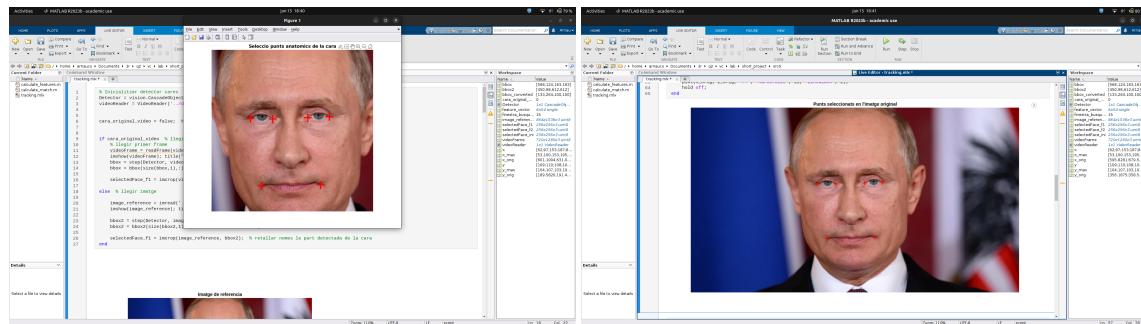
Aquesta imatge de referència m'agrada perquè té una iluminació molt diferent, així es pot comprovar que el meu sistema és robust als canvis d'iluminació. L'únic detall a comentar, és la part exterior del ull dret. I el que encara no he comentat, però sempre m'ha detectat, bé és la part interna de l'ull.

Putin:





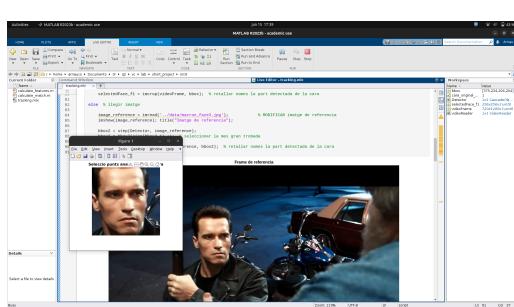
Al utilitzar el primer frame, el matxing és perfecte. Només s'ha de tenir en compte que els punts seleccionats estan en la cara esclada, i s'ha de transformar aquests punts al frame sense escalar.



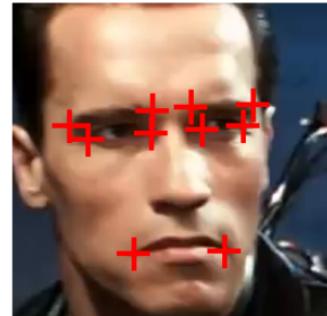
Com era d'esperar, el matxing és una mica pitjor que en el cas anterior. El major detall poc precís és en l'ull dret, a la part de l'esquerra.

Arnold:

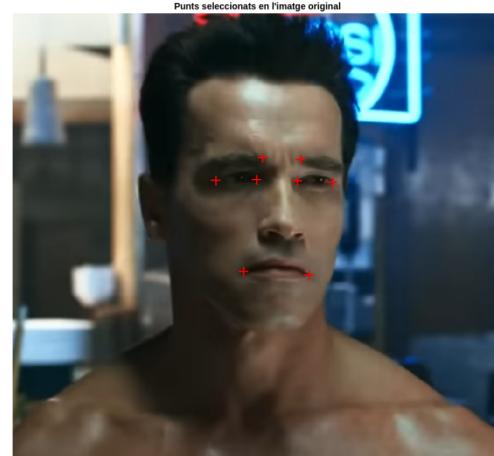
Ara amb un fragment de la pel·lícula Terminator 2, anem a detectar els punts anàtomics d'una màquina poc expressiva.



Selecció punts anàtomics de la cara



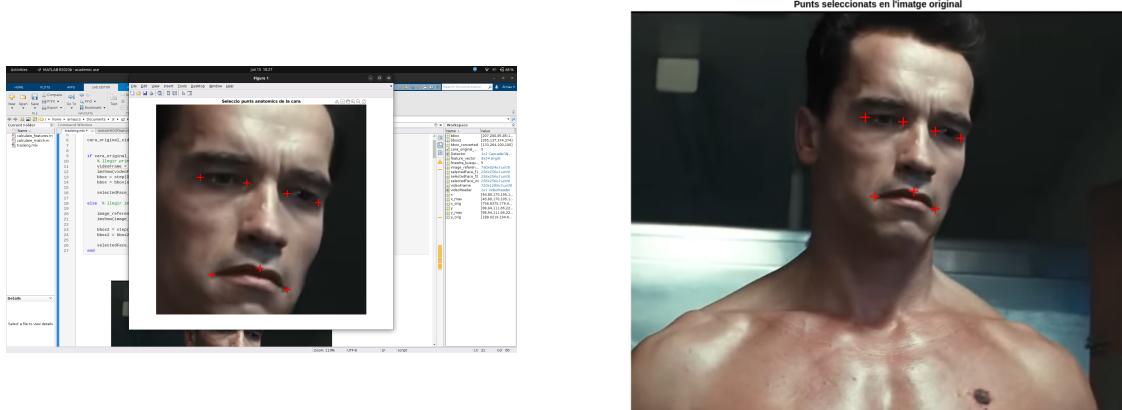
En aquest cas, he seleccionat el que considero els punts més complets, incloent en aquest cas els extrems de les celles, l'únic canvi és que ara el càlcul del matxing inicial tardarà molt més en els altres casos, però al ser la imatge del primer frame, ha funcionat com toca.



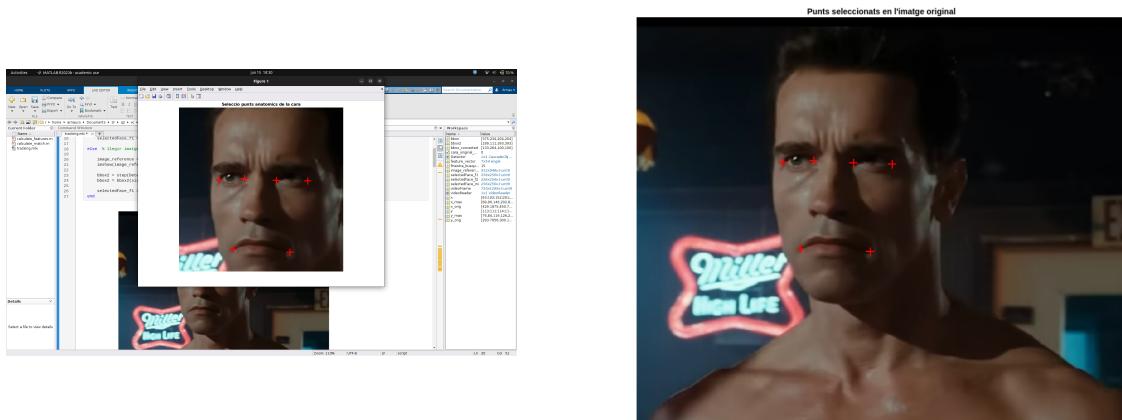


En aquest cas el matxing no ha estat molt bo, el llavi a la part dreta està bastant a baix, i sobretot, els ulls, els ha posat a les celles. Justament en aquesta imatge de test havia seleccionat només la part de la cella més propera al nas i justament, el matxing ha anat tot a les celles.

Aquest frame, és d'una part d'on el Terminator sent una mica d'arrepentiment, i els llavis estan decantats cap avall, i a més he marcat el seu punt central d'aquests llavis.



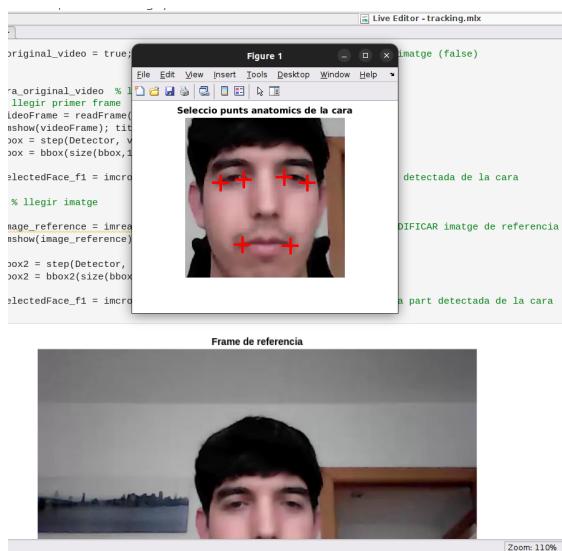
En la detecció dels punts al vídeo, no ha estat molt bona en els ulls, però sí en els llavis.

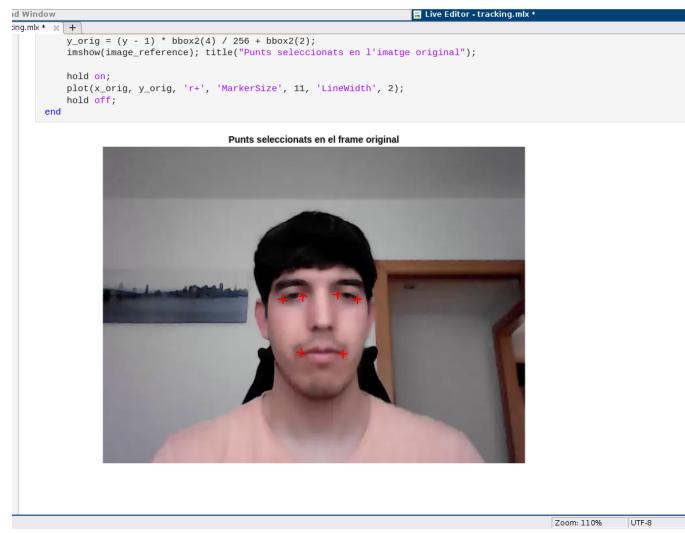


Per acabar la part del test amb l'Arnold, aquest resultat trobo que és acceptable, i que en els continuats frames, es va arreglant poc a poc el petit marge d'error del matxing inicial.

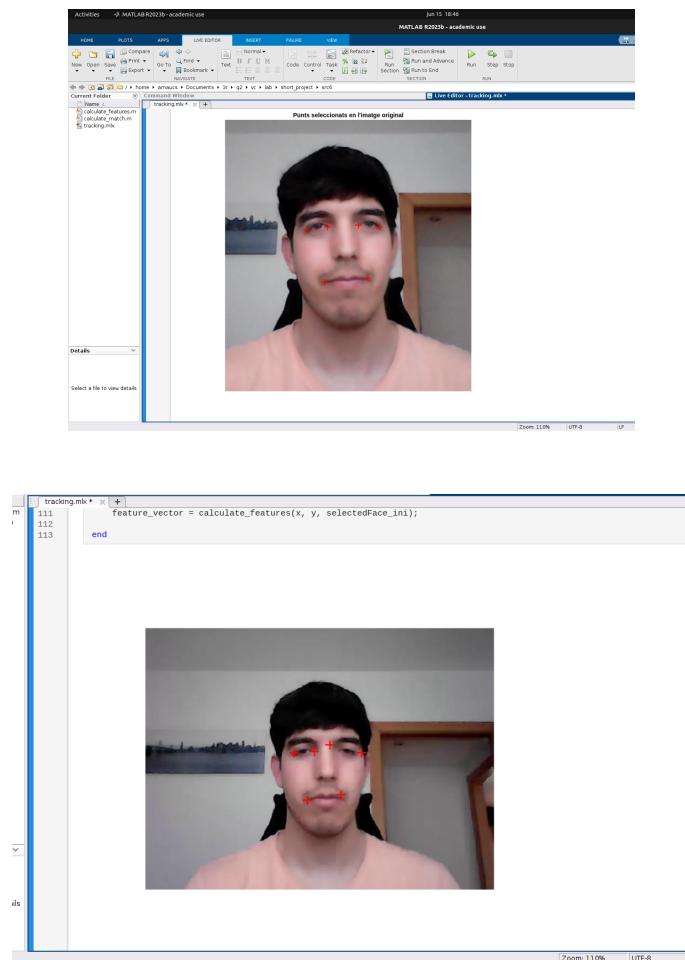
Arnaud Claramunt:

A part de personatges públics, vaig voler veure si funcionaria també amb la meva pròpria cara, per això vaig fer un vídeo molt curt. En el primer test, la imatge és del frame original, i el segon, una imatge que considerava interessant, al pausar el vídeo i fer la captura.





Com tots els casos vistos anteriorment, el frame inicial, funciona perfecte. I ajuda a tenir un bon punt de partida a la detecció al vídeo.



En aquest cas no ha fet molt bé la part interior del ull dret, però en el frame original miro més a baix i tinc els ulls més tancats, però estic satisfet amb el resultat, amb la imatge de test.

6 Resultats en els Videos Test

Aquesta part juntament amb l'explicació de l'execució està present en el vídeo penyat juntament amb aquest informe.

Els videos utilitzats han estat:

- Discurs Macron
- Discurs Obama
- Discurs Putin
- Arnold Schwarzenegger a la pelicula Terminator 2
- Video de l'autor d'aquest projecte

7 Casos que el programa no funciona

Un cas molt concret és en a l'inici del video del Putin, on per una estranya raó no troba la seva cara i per tant els punts detectats estan a la part inferior del vídeo. M'ha sorprès, ja que amb els altres polítics funcionaba bé i amb el Putin pot ser que sigui cosa de la detecció de cares, ja que a classe ja vam veure també que detectava en alguns moments una cara a la columna de la part dreta.

Algun error que passa alguns cops, és que en algun frame no es detecti bé la cara i mostri els punts una mica allunyats, però després es recupera. Es pot veure per exemple en un frame molt avançat del video de l'Obama. La manera que vaig intentar solucionar-ho va ser de totes les bounding boxes que detecta amb les cares, quedar-me amb la més gran, ja que crec que és la més probable que sigui la de la cara. Però no ha solucionat del tot aquest detall.

En algun cas en versions intermitges del desenvolupament del codi, les imatges test al seleccionar punts, no feien bon matxing i per exemple si era comisura exterior del ull, es quedava a la galta, i d'allà mai arribava al ull. Però al millorar el matching i sobretot el càlcul dels features, ja es va anar solucionant.

De manera similar que l'apartat anterior, en el video adjunt, també es mostren resultat incorrectes o amb petits errors. Observant moments on potser en un frame deixa de detectar la cara.

7.1 Problemes trobats i les solucions

Pel que fa més al codi i programari, el MATLAB amb el meu portatil que té Ubuntu, va molt lent. També em sortia un Warning de deshabilitar funcions avançades de rendering, suposo perquè no tinc tarjeta gràfica. I fent el porjecte s'hem va penjar el Matlab uns 4 cops i va tancar inesperadament.

El principal desavantatge i aspecte que m'ha retressat més és el temps d'espera per veure els resultats d'un petit canvi fet. Cada cop que canviava alguna cosa, havia de tornar a seleccionar els punts, anar executant cada zona, i després la part del tracking al vídeo. Que va increïblement lent, fa com un frame cada 4 o 5 segons. La solució va ser paciència i reduir la finestra de cerca del matching en casos que tardava encara més.

8 Comparació amb mètodes existents

En aquesta secció, compararem el nostre sistema de detecció de punts anatòmics de la cara amb alguns mètodes existents reconeguts, com Mediapipe de Google, el mètode Swift, i el mètode Harris.

El Mediapipe de Google és una eina potent que utilitza models de xarxes neuronals per a la detecció i seguiment de punts clau de la cara. Mediapipe ofereix una precisió i robustesa elevades gràcies a la seva capacitat per processar informació en temps real i ajustar-se a diverses condicions d'il·luminació i orientació de la cara. Tot i això, el seu ús pot ser limitat per la seva complexitat i els requisits de comput en el maquinari.

El mètode Swift és conegut per la seva velocitat i eficiència en la detecció de característiques. Utilitza una aproximació basada en la localització de punts d'interès i la seva descripció mitjançant vectors de característiques robustos. Tot i que Swift és molt ràpid, pot no ser tan precís com altres mètodes en situacions de condicions variades d'il·luminació i angles de la cara.

El mètode Harris és un dels mètodes clàssics de detecció de punts d'interès, utilitzat àmpliament en visió per computador. Aquest mètode es basa en el càcul de l'autocorrelació de les intensitats dels píxels per identificar els punts de la imatge que presenten canvis significatius en diverses direccions. El mètode Harris és simple i efectiu, però pot ser susceptible als canvis d'il·luminació i sovint requereix una post-processament addicional per millorar la seva robustesa.

En comparació amb aquests mètodes, el meu sistema de detecció de punts anatòmics de la cara està dissenyat per ser personalitzable segons les necessitats específiques de l'usuari. Tot i que pot no assolir la precisió i la robustesa de Mediapipe, la simplicitat del nostre mètode ofereix avantatges en termes de facilitat d'implementació i ajustament als punts d'interès seleccionats manualment per l'usuari. A més, em centro en l'ús de descriptors de color HSV per minimitzar els efectes de les variacions d'il·luminació, oferint una solució equilibrada entre complexitat i efectivitat.

En resum, cada mètode té els seus avantatges i inconvenients, i la selecció d'un mètode particular depèn dels requisits específics del projecte i dels recursos disponibles. I en el meu cas no usar alguns mètodes ja existents, i m'ha agradat experimentar amb varis i tenir el suspens de veure quin va millor.

9 Annex (Codi)

Estructura dels codis està dividida en 3 fitxers, el del codi principal, el de la funció calcular vector característiques i el de la funció de calcular el matxing. Els posaré en aquest ordre. Cada bloc de codi es un section break separat.

Codi Principal

```
1 % Detectar elements anatomicos de la cara:
2
3 % Inicialitzar detector de cares
4 Detector = vision.CascadeObjectDetector('FrontalFaceLBP');
5 videoReader = VideoReader('../data/Obama.webm'); % MODIFICAR nom del video
6
7 % MODIFICAR segons si volem video (true) o imatge (false)
8 cara_original_video = true;
9
10 if cara_original_video % llegir video
11     % llegir primer frame
12     videoFrame = readFrame(videoReader);
13     imshow(videoFrame); title("Frame de referencia");
14     bbox = step(Detector, videoFrame);
15     bbox = bbox(size(bbox,1),:); % seleccionar la bb mes gran trobada
16
17     selectedFace_f1 = imcrop(videoFrame, bbox); % retallar nomes la part detectada
18     % de la cara
19
20 else % llegir imatge
21
22     image_reference = imread('../data/obama_face3.jpg'); % MODIFICAR imatge de
23     % referencia
24     imshow(image_reference); title("Imatge de referencia");
25
26     bbox2 = step(Detector, image_reference);
27     bbox2 = bbox2(size(bbox2,1),:); % seleccionar la mes gran trobada
28
29 end
30
31 % Reescalar (sempre imatge 256x256)
32 selectedFace_f1 = imresize(selectedFace_f1, [256,256]);
33 imshow(selectedFace_f1); title("Seleccio punts anatomicos de la cara")
34
35 hold on;
36 [x, y] = getpts; % guardar punts marcats
37 x = round(x); y = round(y);
38
39 % mostrar creus vermelles als punts
40 plot(x, y, 'r+', 'MarkerSize', 20, 'LineWidth', 3, 'Color', 'red');
41
42 hold off;
43
44 % Dibuixar punts al frame o imatge original
45
46 if cara_original_video
47     % Mapejar els punts seleccionats en les coordenades originals
48     x_orig = (x - 1) * bbox(3) / 256 + bbox(1);
49     y_orig = (y - 1) * bbox(4) / 256 + bbox(2);
50     imshow(videoFrame); title("Punts seleccionats en el frame original");
51
52     hold on;
53     plot(x_orig, y_orig, 'r+', 'MarkerSize', 11, 'LineWidth', 2);
54     hold off;
55
56 % Vector de caracteristiques dels punts anatomicos
57 % Calcul dels features en una funcio, a partir dels punts marcats i la cara
58 actual
```

```

15 feature_vector = calculate_features(x, y, selectedFace_f1);
16
17 else % tenim imatge de referencia
18 % Mapejar els punts seleccionats en les coordenades originals de l'imatge
19 x_orig = (x - 1) * bbox2(3) / 256 + bbox2(1);
20 y_orig = (y - 1) * bbox2(4) / 256 + bbox2(2);
21 imshow(image_reference); title("Punts seleccionats en l'imatge original");
22
23 hold on;
24 plot(x_orig, y_orig, 'r+', 'MarkerSize', 11, 'LineWidth', 2);
25 hold off;
26 end

```

```

1 % NOMES EXECUTAR seccio si tenim imatge de referencia, sino no cal
2
3 % continuacio de la part anterior si hi havia imatge de referencia
4 if not(cara_original_video)
5 % Fer matching a la cara del primer frame del video
6 videoFrame = readFrame(videoReader);
7
8 bbox = step(Detector, videoFrame);
9 bbox = bbox(size(bbox,1),:); % seleccionar la mes gran trobada
10
11 selectedFace_ini = imcrop(videoFrame, bbox); % retallar nomes la part
12 detectada de la cara
13
14 % Reescalar imatge cara
15 selectedFace_ini = imresize(selectedFace_ini, [256,256]);
16
17 % Buscar matching de punts segons els features
18
19 finestra_busqueda = 15; % finestra a mirar al voltant dels punts seleccionats,
20 % per optimitzar la cerca
21
22 % Vector de caracteristiques dels punts anatomicos de la imatge de
23 % referencia amb els punts seleccionats
24 feature_vector = calculate_features(x, y, selectedFace_f1);
25
26 % Coordenades on hi haura els punt del pixel amb similitud maxima als features
27 [x_max, y_max] = calculate_match(finestra_busqueda, selectedFace_ini, x, y,
28 feature_vector);
29
30 % dibuixar punts al frame original
31
32 % Mapejar els punts seleccionats en les coordenades originals
33 x_orig = (x_max - 1) * bbox(3) / 256 + bbox(1);
34 y_orig = (y_max - 1) * bbox(4) / 256 + bbox(2);
35
36 imshow(videoFrame);
37
38 hold on;
39 plot(x_orig, y_orig, 'r+', 'MarkerSize', 11, 'LineWidth', 2);
40 hold off;
41
42 % actualitzar punts pel seguent frame
43 x = x_max;
44 y = y_max;
45
46 % tornar a calcular vector features, pero ara sobre el primer frame ja
47 % detectat els punts on toquen
48 feature_vector = calculate_features(x, y, selectedFace_ini);
end

```

```

1 % Tracking en el video
2
3 while hasFrame(videoReader)
4     videoFrame = readFrame(videoReader);
5     bbox = step(Detector, videoFrame);
6
7 if isempty(bbox)
8     continue; % saltar el frame si no detecta cap cara
9 end
10
11
12 bbox_converted = bbox(size(bbox,1),:); % quedarse amb el bounding box mes gran
13 % detectat
14
15 % Seleccionar cara:
16 selectedFace_f2 = imcrop(videoFrame, bbox_converted);
17
18 % Reescalar imatge cara
19 selectedFace_f2 = imresize(selectedFace_f2, [256,256]);
20 % Buscar matching de punts segons els features
21
22 finestra_busqueda = 5; % finestra a mirar al voltant dels punts seleccionats,
23 % per optimitzar la cerca
24
25 % Coordenades on hi haura els punt del pixel amb similitud maxima als features
26 [x_max, y_max] = calculate_match(finestra_busqueda, selectedFace_f2, x, y,
27 feature_vector);
28
29 % dibuixar punts al frame original
30
31 % Mapejar els punts seleccionats en les coordenades originals
32 x_orig = (x - 1) * bbox(3) / 256 + bbox(1);
33 y_orig = (y - 1) * bbox(4) / 256 + bbox(2);
34
35 imshow(videoFrame);
36
37 hold on;
38 plot(x_orig, y_orig, 'r+', 'MarkerSize', 11, 'LineWidth', 2);
39 hold off;
40 pause(0.00005); % petita pausa per veure el frame actual del video
41
42 % actualitzar punts per el seguent frame
43 x = x_max;
44 y = y_max;
45 end

```

Funció Calcular Vector Features

```
1 % Funció de codi porpi que detecta varis features en un frame donat i els
2 % retorna en un vector
3
4 function [feature_vector] = calculate_features(x, y, videoFrame)
5
6 feature_vector = []; % vector que contindra a cada columna, un component dels
7 % features i cada fila un punt seleccionat
8
9 numero_punts = length(x);
10
11 punts_voltant = 8; % numero de punts adjacents a mirar
12
13
14 hsvFrame = rgb2hsv(videoFrame);
15 % extreure components H i S
16 H = hsvFrame(:,:,1);
17 S = hsvFrame(:,:,2);
18
19 escala_grisos = rgb2gray(videoFrame);
20
21 % Iterar pels punts seleccionats
22 for i = 1:numero_punts
23     center_y = y(i);
24     center_x = x(i);
25
26     % Feature 1: HSV extreure la component H i S del punt seleccionat
27     H_center = H(center_y, center_x);
28     S_center = S(center_y, center_x);
29
30     % Feature 2: HOG, en una regio petita al voltant punt
31
32     window_size = [16 16]; % ajustar el valor
33     ROI = imcrop(escala_grisos, [center_x - window_size(1)/2, center_y -
34     window_size(2)/2, ... window_size(1)-1, window_size(2)-1]);
35
36     HOG_feature = extractHOGFeatures(ROI);
37
38     % Iterar pels veins adjacents
39     offsets = [-1 0; 1 0; 0 -1; 0 1; -2 0; 2 0; 0 -2; 0 2]; % Top bottom left
40     right, Top2 bottom2 left2 right 2
41
42     neighbor_features = zeros([1 punts_voltant*2]);
43
44     for j = 1:punts_voltant
45         neighbor_y = center_y + offsets(j, 1);
46         neighbor_x = center_x + offsets(j, 2);
47
48         % Calcular H i S dels adjacents
49         H_neighbor = H(neighbor_y, neighbor_x);
50         S_neighbor = S(neighbor_y, neighbor_x);
51
52         neighbor_features(2*j - 1) = H_neighbor;
53         neighbor_features(2*j) = S_neighbor;
54     end
55
56     % afegir totes les features al vector
57     feature_vector = [feature_vector; H_center, S_center, neighbor_features
58     (:,:), HOG_feature];
59
60 end
```

Funció Calcular Matching

```
1 % Funcio que donada una finestra de cerca , la imatge de la cara , les
2 % coordenades dels punts del frame anterior i el vector de features . Retorna les
3 % coordenades x,y dels punts amb match mes elevat .
4
5 function [x_max, y_max] = calculate_match(finestra_busqueda , face , x , y ,
6 feature_vector )
7
8 % Nombre de punts a considerar
9 numero_punts = length(x);
10
11 % Coordenades dels punt del pixel amb similitud maxima als features
12 x_max = zeros(1, numero_punts);
13 y_max = zeros(1, numero_punts);
14
15 % Inicialitzar el valor minim de la distancia amb un valor gran
16 min_distance = inf;
17
18 % Dimensions de la imatge
19 [height , width] = size(face );
20
21 % la metrica que utilitzar per comparar els colors del feature 1 es la
22 % distancia Euclideana
23
24 % recorrer elements en una finestra al voltant de les coordenades del punt
25 % seleccionat i quedar-se amb el maxim
26
27 for p = 1:numero_punts
28     % Inicialitzar el valor minim de la distancia amb un valor gran
29     min_distance = inf;
30
31     % Coordenades inicials del punt
32     x_inicial = x(p);
33     y_inicial = y(p);
34
35     % Vector de feature original per aquest punt
36     feature_original = feature_vector(p, :);
37
38     % Recorrer elements en una finestra al voltant de les coordenades del punt
39     % seleccionat i quedar-se amb el maxim
40     for i = -finestra_busqueda:finestra_busqueda
41         for j = -finestra_busqueda:finestra_busqueda
42
43             % Coordenades del pixel actual
44             x_actual = x_inicial + i;
45             y_actual = y_inicial + j;
46
47             % Comprovar si les coordenades son dins dels limits de la imatge
48             if x_actual > 0 && x_actual <= width && y_actual > 0 && y_actual <=
49 height
50                 % Obtenir el vector de features del pixel actual
51                 current_feature = calculate_features(x_actual , y_actual , face );
52
53                 % Calcular la distancia euclidiana entre el vector de features
54                 % actual i el vector de features original
55                 distance = norm(current_feature - feature_original);
56
57                 % Si la distancia es menor que la minima distancia trobada fins
58                 % ara , actualitzar les coordenades i la minima distancia
59                 if distance < min_distance
60                     min_distance = distance;
61                     x_max(p) = x_actual;
62                     y_max(p) = y_actual;
63                 end
64             end
65         end
66     end
67 end
68 end
69 end
```