

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Sessió 6

INFORME

VISIÓ PER COMPUTADOR

Autors

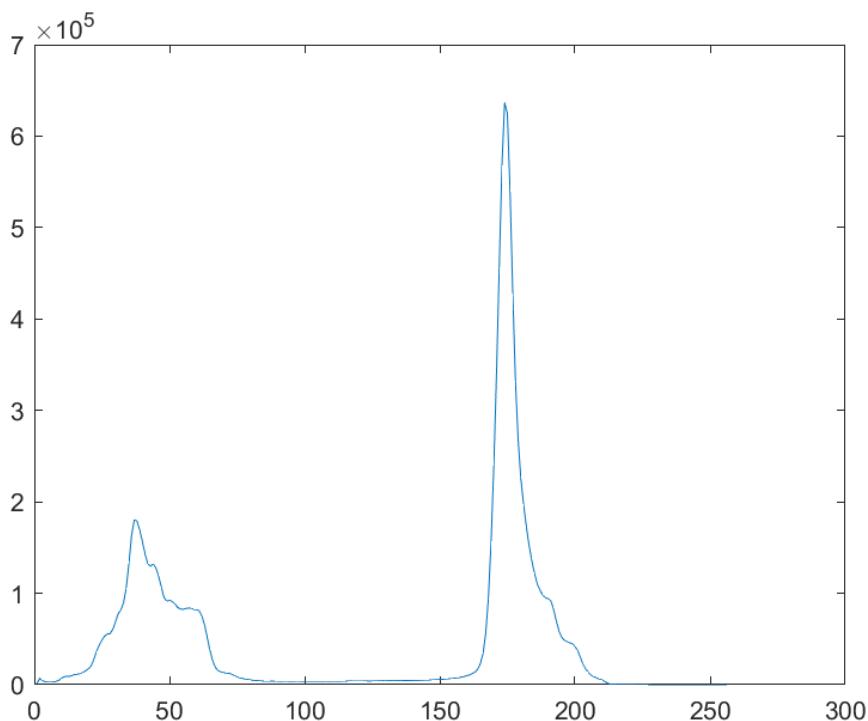
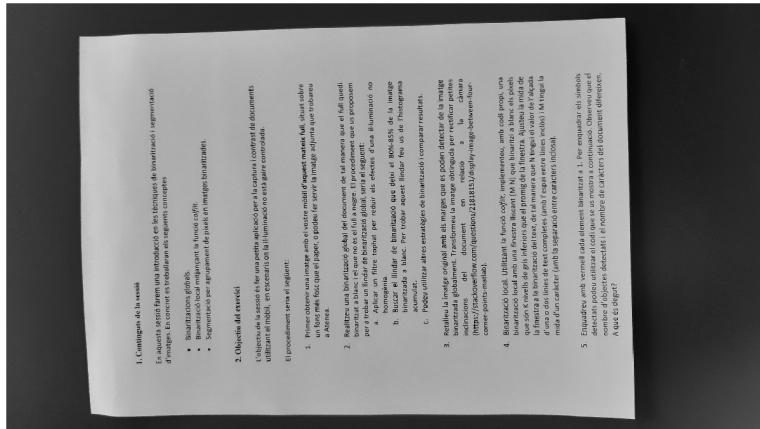
DAVID LATORRE
ADRIÀ AUMATELL

- Pas 1: Obtenció de la imatge

Primer de tot, carregarem la imatge de la fotografia del full que se'ns ha facilitat i n'obtenim l'histograma, que farem servir més endavant.

Codi:

```
I = rgb2gray(imread('full.jpg'));
imshow(I);
h = imhist(I);
figure
plot(h);
```

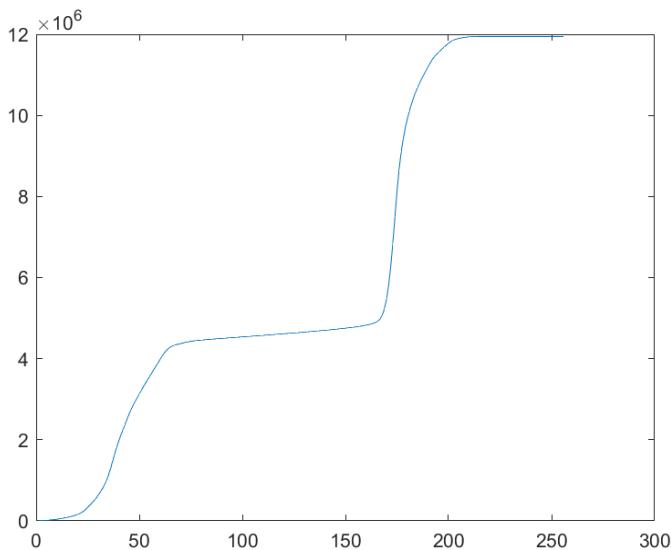


- Pas 2: Binarització per àrea

En aquest apartat intentarem binaritzar la imatge obtinguda a l'apartat anterior. Per poder fer-ho correctament ens cal trobar un bon llindar de binarització. Primer calculem l'histograma acumulat de la imatge.

codi:

```
ha = cumsum(h);
figure
plot(ha);
```



Seguidament, binaritzem per àrea, definint el % de blanc que volem que romangui a la imatge. Per obtenir uns resultats satisfactoris, és a dir, per tal que només quedi a blanc el paper, binaritzarem deixant només un 40% de la imatge a blanc.

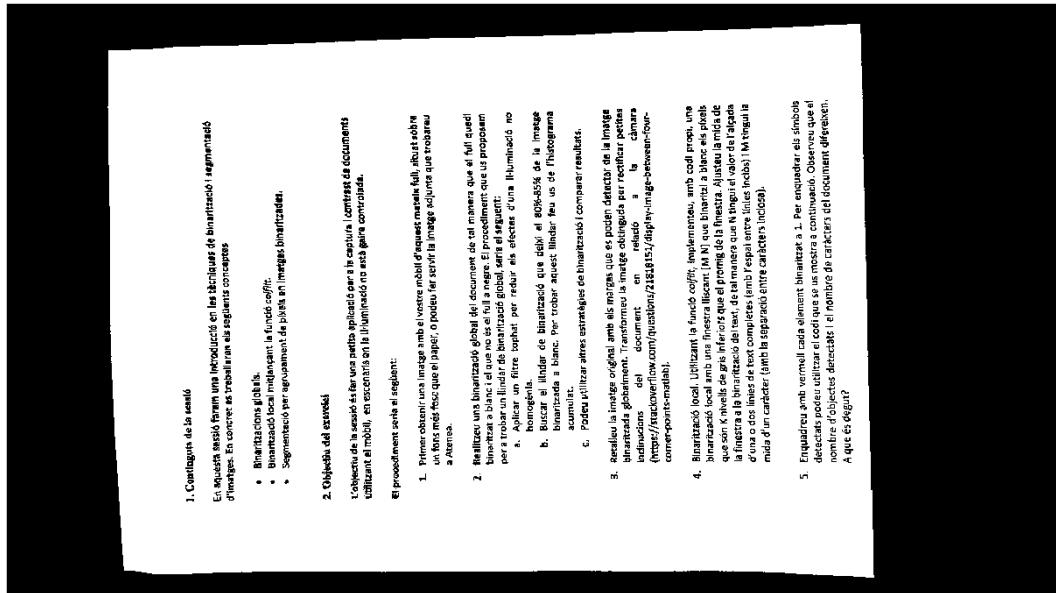
El procediment seguit consisteix en, un cop definida l'àrea a binaritzar, trobar el primer nivell de gris que tingui una àrea major que la definida a l'histograma acumulat.

codi:

```
[files, columnes] = size(I);
mida = files*columnes;
Area = 0.4 * mida; % tan per cent de la imatge a blanc
b = ha > Area; %vector binar amb 0 i 1.
llindar = find(b,1); %busca el primer element a 1
B = I > llindar;

figure
imshow(B);
```

Resultat obtingut:



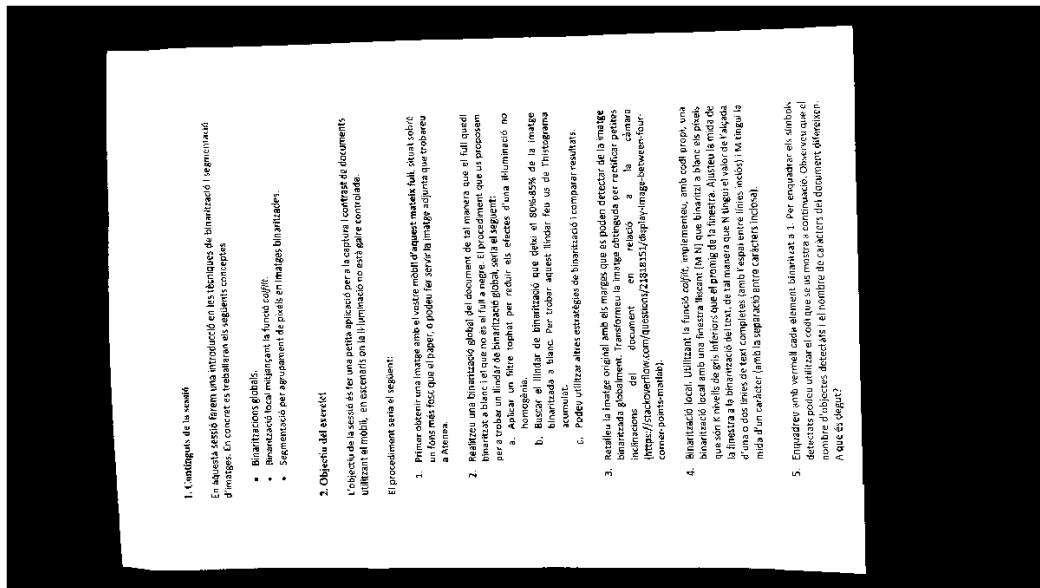
Com es pot observar, els resultats son força satisfactoris, però encara existeixen petites regions negres que no formen part de cap caràcter.

Anem a provar ara una binarització més senzilla. Si ens fixem amb l'histograma (no acumulat) obtingut a l'apartat anterior, podem veure que hi ha dues regions de nivells de gris molt clares a la imatge, corresponents al full i a les lletres i al fons, que és precisament el que volem separar. Agafem, doncs, un nivell de gris que estigui entre aquestes dues zones, per exemple 120.

codi:

```
C = I > 120;  
figure  
imshow(C);
```

Resultat:



- Binaritzacions globals.
- Binarització local mitjançant el seu entorn.
- Segmentació per agrupació d'imatges.

2. Objectiu del exercici

L'objectiu de la sessió és fer una petit aplicació per a la següent tòcnica de documents ultranits al mòbil, en escenaris utilitzant el mòbil, en escenaris

El procediment seria el següent:

1. Primer obtenir una imatge un fons més fosc que el i a Atenea.
 - a. Aplicar un filtro top-hat per a trobar un llindar d'imatge.
 - b. Buscar el llindar de binarització a blanc.
2. Realitzeu una binarització binaritzat a blanc i el que no es el full negre. El procediment que us proposem per a trobar un llindar de binarització global, sent el següent:
 - a. Aplicar un filtro top-hat per a trobar un llindar d'imatge.
 - b. Buscar el llindar de binarització que dóna el 50% de la imatge binaritzada a blanc. Per trobar aquest llindar feu us de l'histograma acumulatiu.
 - c. Podeu utilitzar altres estratègies de binarització com paràmetres resultats.

Els resultats també son força satisfactoris. Fins i tot es podria dir que amb el binaritzat simple la imatge resultant és més nítida i no s'observen les petites regions negres que apareixen a l'altre binaritzat. En tot cas, en la resta de passos s'agafa com a base el binaritzat per àrea.

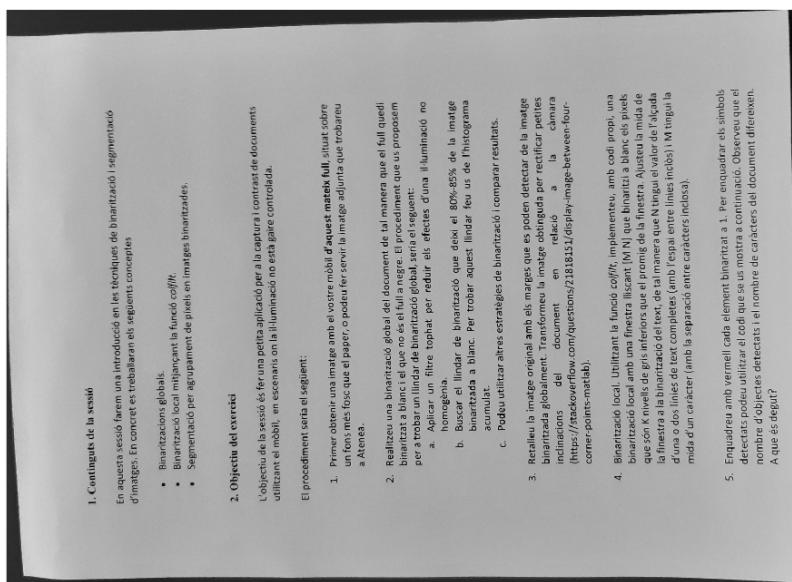
- Pas 3: Retallat de marges i correcció de la inclinació

Per tal de poder detectar correctament els caràcters cal que rectifiquem la inclinació del document. Primer de tot, obtindrem els valors x i y màxims i mínims del full i els utilitzarem per enquadrar-lo en una nova imatge.

codi:

```
[rows, columns] = find(B);
row1 = min(rows);
row2 = max(rows);
col1 = min(columns);
col2 = max(columns);

RI = I(row1:row2, col1:col2, :);
figure
imshow(RI);
```

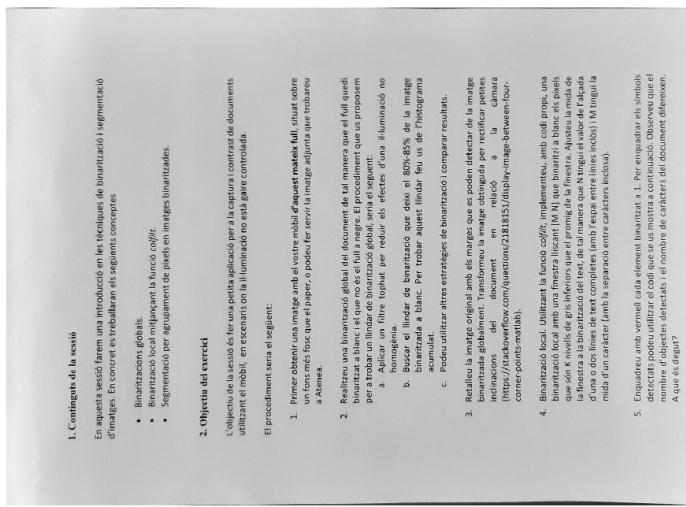


Ara ja hem eliminat bona part dels marges, però, com es pot observar, el full està inclinat. Per arreglar-ho, ens caldrà saber la posició dels quatre vèrtexs del full. Mitjançant la funció Data Tips de Matlab podem obtenir-los fàcilment de forma manual. Seguidament, només cal aplicar les instruccions que es mostren en el fòrum indicat i ja tindrem la inclinació rectificada.

codi:

```
[filesR, columnesR] = size(RI);
fixedPoints=[1 1;columnesR 1; columnesR filesR; 1 filesR]; %(x,y) coordinate
movingPoints=[1 113;3305 1;3361 2426;76 2392];

TFORM = fitgeotrans(movingPoints,fixedPoints,'projective');
RE=imref2d(size(RI),[1 size(RI,2)],[1 size(RI,1)]);
imgTransformed=imwarp(RI,RE,TFORM,'OutputView',RE);
figure
imshow(imgTransformed,[]);
```



Tot i que el resultat és força bo, encara queda algun rastre dels marges, sobretot a la part inferior dreta. Segurament, això passa perquè el full no estava completament pla en el moment de fer la fotografia i, en conseqüència, el paper no té una forma rectangular perfecta.

- Pas 4: Binarització local

En aquest pas binaritzarem la imatge localment. És a dir, per tal de determinar si un pixel serà blanc o negre, observarem els valors dels pixels que estiguin a dins de la finestra lliscant centrada al pixel que estem avaluant, i a partir de tots aquests determinarem si el pintem blanc o negre.

Per a aquest determinat cas el que ens interessa son els caràcters del text, i per tant volem que aquells pixels que facin referència a caràcters siguin pintats de color blanc, mentre que aquells que no siguin pintats de color negre.

Com que la imatge que tenim actualment té la fulla de color blanc, i el text escrit en color negre, pintarem un pixel de la imatge de color blanc només quan el valor d'aquest sigui inferior al valor promig de tots els pixels de la finestra lliscant - k , on k és un natural que fa referència a una quantitat de nivells de grisos.

És d'esperar que si triem un valor k molt baix, els caràcters ens quedin molt gruixuts, de forma que es puguin acabar juntant entre ells fent un càlcul incorrecte de les components connexes i per tant del reconeixement de caràcters... I, si triem un valor k molt alt, per altre banda els caràcters ens quedarien molt prims, i alguns fins i tot podrien desaparèixer o dividir-se, de forma que el reconeixement d'aquests també seria difícil. Es tracta, per tant, de trobar el valor ideal de k .

Abans però de binaritzar la imatge, perquè la binarització sigui eficaç ens hem d'assegurar que la imatge que tenim té una correcta il·luminació. Observem doncs, la imatge i els seus histogrames (histograma normal, i histograma acumulat):

1. Continguts de la sessió
En aquesta sessió farem una introducció en les tècniques de binarització i segmentació d'imatges. En concret es treballaran els següents conceptes

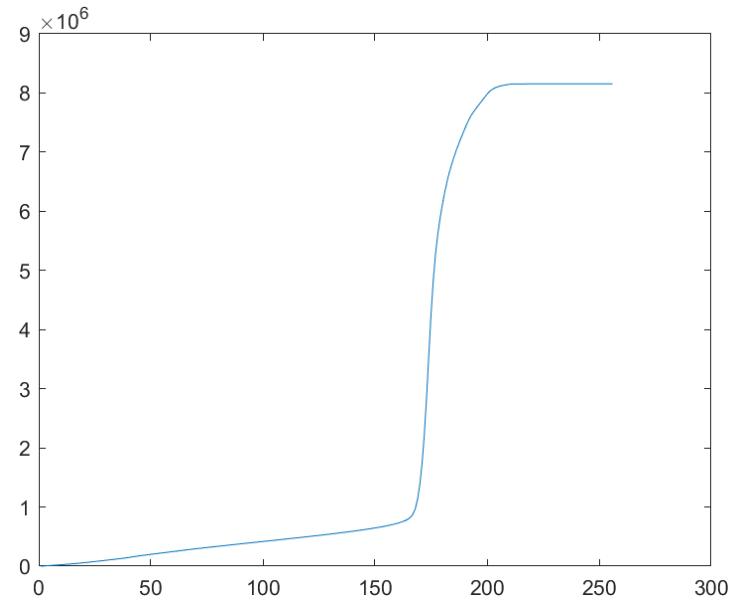
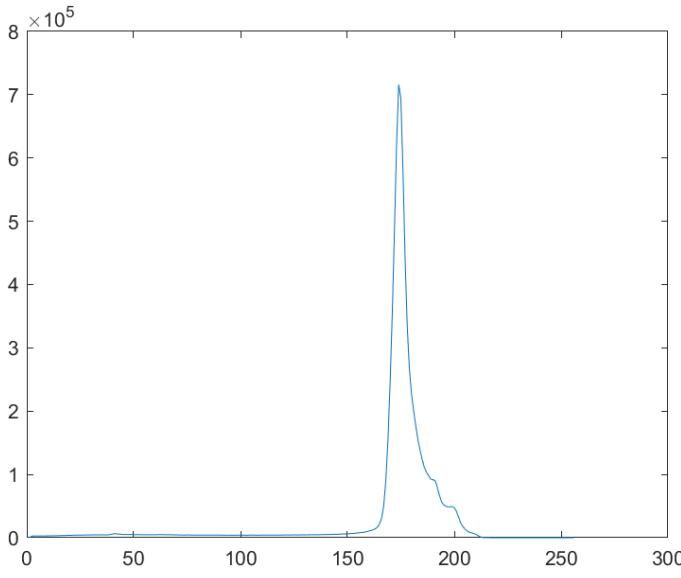
- Binaritzacions globals.
- Binarització local mitjançant la funció `imfilter`.
- Segmentació per agrupament de pixels en imatges binaritzades.

2. Objectiu del exercici

L'objectiu de la sessió és fer una petit aplicació per a la capura i contrastar de documents utilitzant el mòbil, en escenaris on la il·luminació no està gaire controlada

El procediment sera el següent:

1. Primer obtenirem un imatge amb el nostre mòbil d'aquest mateix full, situat sobre un fons més fos, que si possem fer servir la imatge adjunta que trobareu a Arees.
2. Obtenire una binarització global del document de la manera que el feu quedí binaritzat a blanc el que no és el la negre. El procediment que us proposem per a fer-ho es l'aplicació d'un mètode binarització global, cosa el següent:
 - a. Fer una copia del document per redó, dona il·luminació més homogènia.
 - b. Binaritzar el llinatge de binarització que deixi el 20% 85% de la imatge binaritzada a blanc. Per trobar aquest valor feu us de l'histograma acumulat.
 - c. Podeu utilitzar altres estratègies de binarització i comparar resultats.
3. Redueix la imatge original amb els margens que creuades detecta de la seva binarització global. Transformeu la imatge a escala de colors i segmenteu-la en els diferents components de la imatge. (veure <https://stackoverflow.com/questions/218151/display-image-between-boundaries>).
4. Binarització local utilitzant la funció `gofft`. Implementeu, amb codi propi, una binarització local mitjançant la funció `gofft` de la qual us serveu per a fer servir la imatge de la binarització del text, de la qual fareu que N sigui el valor de l'activitat d'una o dos files de text completes (amb l'espai entre caràcters inclos). Mtingula mida a un caràcter amb la separació entre caràcters inclos.
5. Enquadrateu amb vermell cada element binaritzat a 1. Per enquadrateu els symbols detectats podeu utilitzar el codi que us mostra a continuació. Obsereu que el nombre d'objectes detectats i el nombre de caràcters del document difereixen. A què es deu?



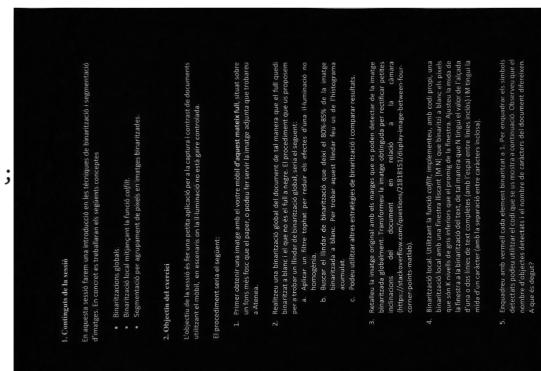
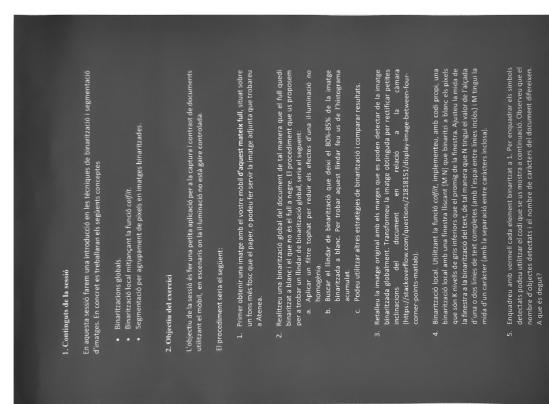
Només veient la imatge ja observem que la tonalitat dels blancs del full és variable. Aquesta incorrecta il·luminació també la observem a l'histograma de la imatge.

El que hem de fer, per tant, és arreglar aquesta il·luminació abans de fer la binarització. Això ho podem fer fàcilment amb un top-hat, el qual és quedaria amb objectes petits eliminant els grans, de forma que eliminariem aquesta tonalitat no homogènia de colors blancs. Per realitzar el top-hat, però, primer hem d'invertir la imatge. Llavors apliquem el top-hat, i finalment tornem a invertir la imatge.

Codi:

```
%inversio per tophat  
imgTransformed = 255 - imgTransformed;  
figure  
imshow(imgTransformed);
```

```
%el top-hat
se = strel('disk',25);
imgTransformedfiltered = imtophat(imgTransformed,se);
figure
imshow(imgTransformedfiltered)
```



Ja tenim la il·luminació arreglada. Procedim ara a fer la binarització.

Per fer la binarització farem ús de la funció blkproc. En aquesta funció primerament li haurem de passar el tamany de la finestra lliscant, i la k (el que hem explicat anteriorment). LLavors blkproc cridarà a una certa funció tantes vegades com pixels hi hagi, i aquesta funció calcularà el valor binaritzat del pixel que està avaluant a partir de la k i el promig de tots els pixels de la finestra lliscant, tal com hem explicat anteriorment. Aquesta funció, la qual hem anomenat “myfunc.m”, té la següent forma:

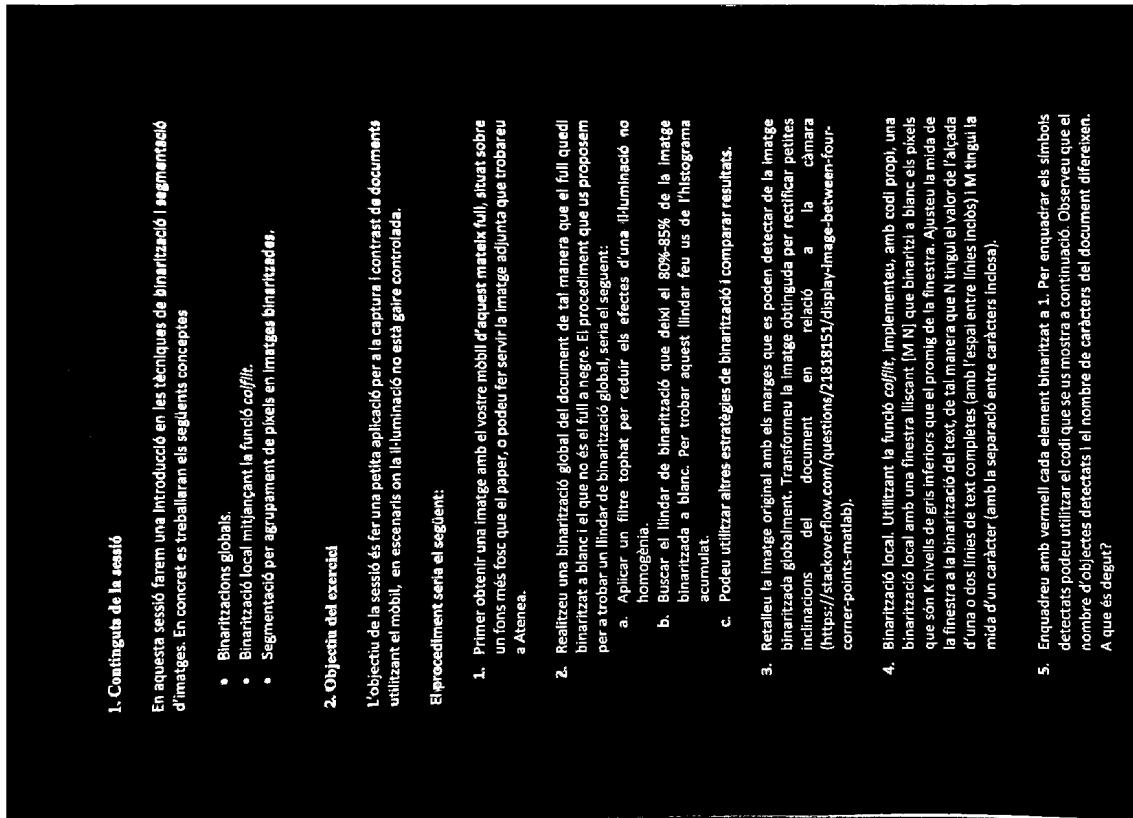
```
function [y] = myfun(x, k)
%y = mean(x, 'all') > k; %binaritzacio global
%binaritzacio local
m = mean(x, 'all');
[f, c] = size(x);
cf = round(f/2);
cc = round(c/2);
y = x(cf,cc) > m;
end
```

Ara ens falta triar quins valors de finestra lliscant, i k, li passarem a la funció blkproc. Principalment, i com que tenim la fulla en horitzontal, hem triat com a dimensions de la finestra lliscant: [25, 70]. És a dir, 25 pixels d'altura, i 70 d'amplada. Hem triat 25 perquè el l'espai aproximant entre un caràcter i el següent (amb espais inclosos), i hem triat 70 perquè és l'espai entre una línia del text i la següent (amb espais inclosos).

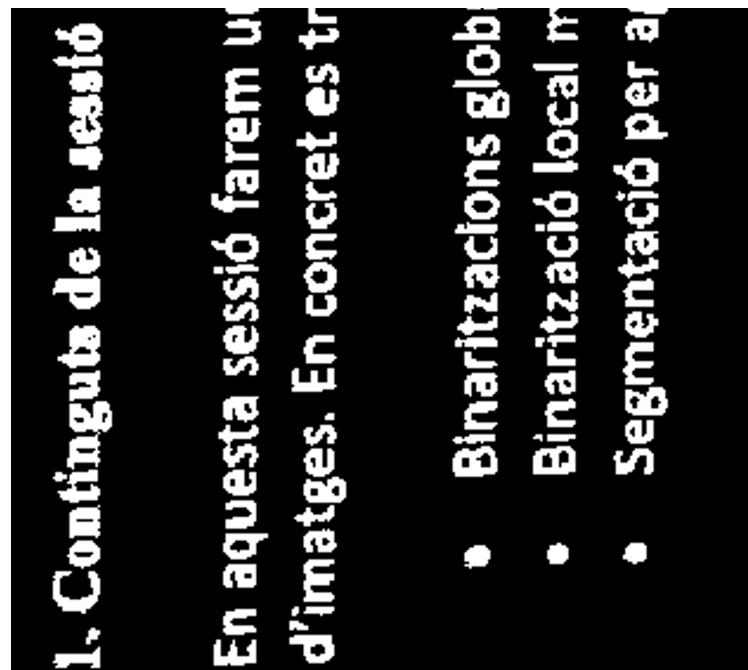
Per que fa al valor k, aquí haurem d'experimentar per tal de trobar el millor.

Fem doncs una primera prova amb la finestra lliscant que hem dit, i $k = 10$.

Output:



Output amb zoom:



A primera vista la binarització no és dolenta. No obstant el valor de k no és l'ideal, ja que podem veure com els caràcters queden molt gruixuts, i fins i tot per exemple les “s” en negreta no es poden reconèixer bé, entre d'altres caràcters.

Triem doncs un valor k més gran. Provem amb k = 40:

Output:

1. Continguts de la sessió

En aquesta sessió farem una introducció en les tècniques de binarització i segmentació d'imatges. En concret es treballaran els següents conceptes

- Binaritzacions globals
- Binarització local mitjançant la funció `coffit`.
- Segmentació per agrupament de pixels en imatges binaritzades.

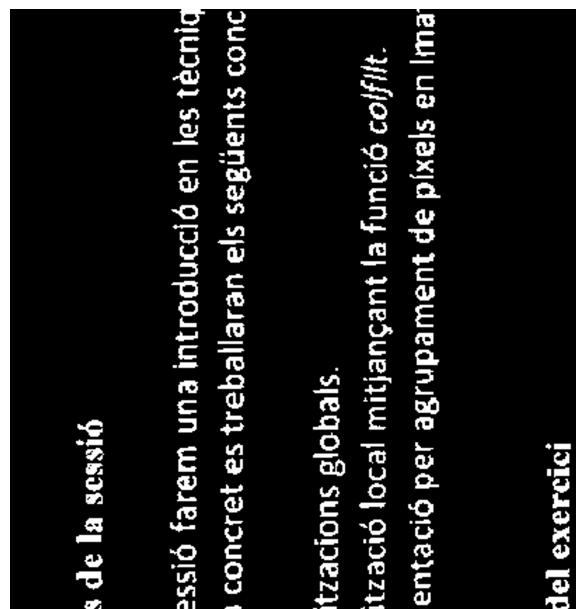
2. Objectiu del exercici

L'objectiu de la sessió és fer una petita aplicació per a la captura i contrast de documents utilitzant el mòbil, en estenarlos on la il·luminació no està gaire controlada.

El procediment seria el següent:

1. Primer obtenir una imatge amb el vostre mòbil d'aquest mateix full, situat sobre un fons més fosc que el paper, o podeu fer servir la imatge adjunta que trobareu a Ateneus.
2. Realitzeu una binarització global del document de tal manera que el full quedi binaritzat a blanc i el que no es el full a negre. El procediment que us proposem per a trobar un líndar de binarització global, seria el següent:
 - a. Aplicar un filtre top hat per reduir els efectes d'una il·luminació no homogènia.
 - b. Buscar el líndar de binarització que deixi el 80%-85% de la imatge binaritzada a blanc. Per trobar aquest líndar feu us de l'histograma acumulat.
 - c. Podeu utilitzar altres estratègies de binarització i comparar resultats.
3. Retalleu la imatge original amb els marges que es poden detectar de la imatge binaritzada globalment. Transformeu la imatge obtinguda per rectificar petites inclinacions del document en relació a la càmera (<https://stackoverflow.com/questions/21818151/display-image-between-four-corner-points-matlab>).
4. Binarització local. Utilitzant la funció `coffit`, implementeu, amb codi propi, una binarització local amb una finestra il·liscant $(M \times N)$ que binaritza a blanc els pixels que son K nivells de gris inferiors que el promig de la finestra. Ajusteu la mida de la finestra a la binarització del text, de tal manera que Vingui el valor de l'alçada d'una o dos línies de text completes (amb l'espai entre línies inclos) i M tingui la mida d'un caràcter (amb la separació entre caràcters inclosa).
5. Enquadreu amb vermell cada element binaritzat a 1. Per enquadrar els símbols detectats podeu utilitzar el codi que se us mostra a continuació. Observeu que el nombre d'objectes detectats i el nombre de caràcters del document difereixen. A què és degut?

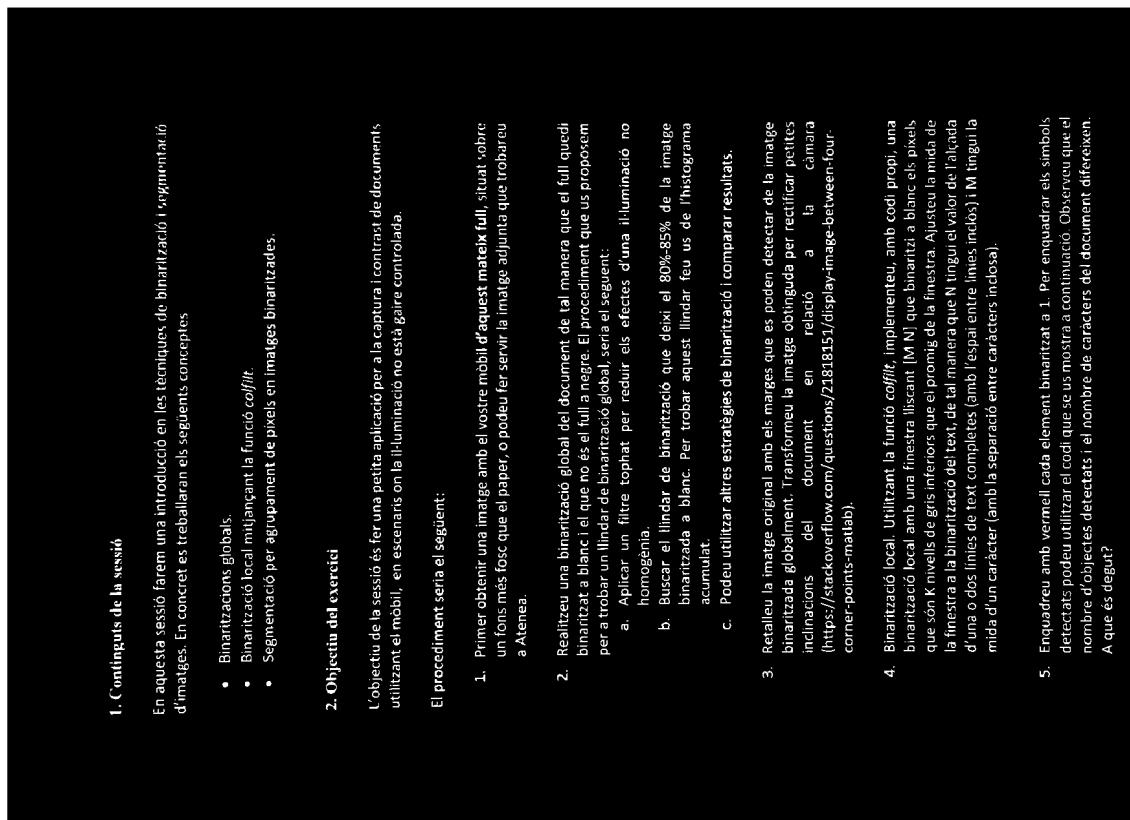
Output amb zoom:



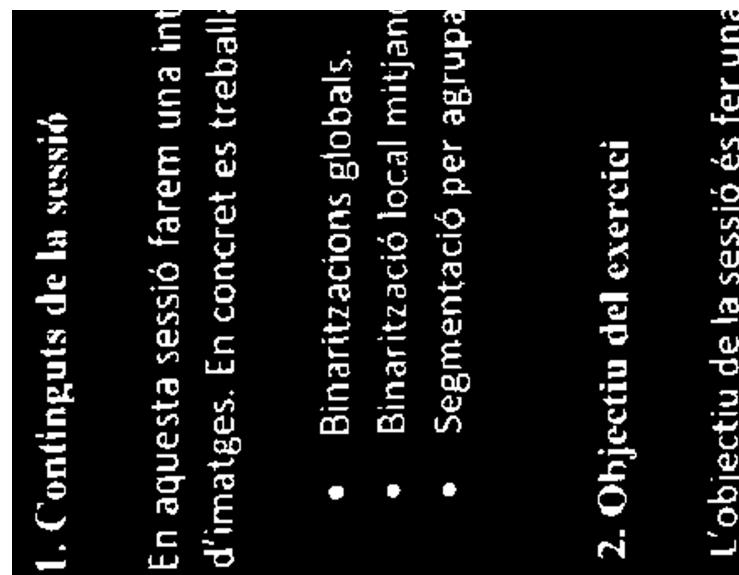
El resultat millora respecte la primera prova, i ja es bastant bo. No obstant encara podem millorar la k una mica més, ja que observem que alguns caràcters no es poden acabar de reconèixer bé perquè segueixen essent gruixuts, així com accents.

Provem amb $k = 60$

Output:



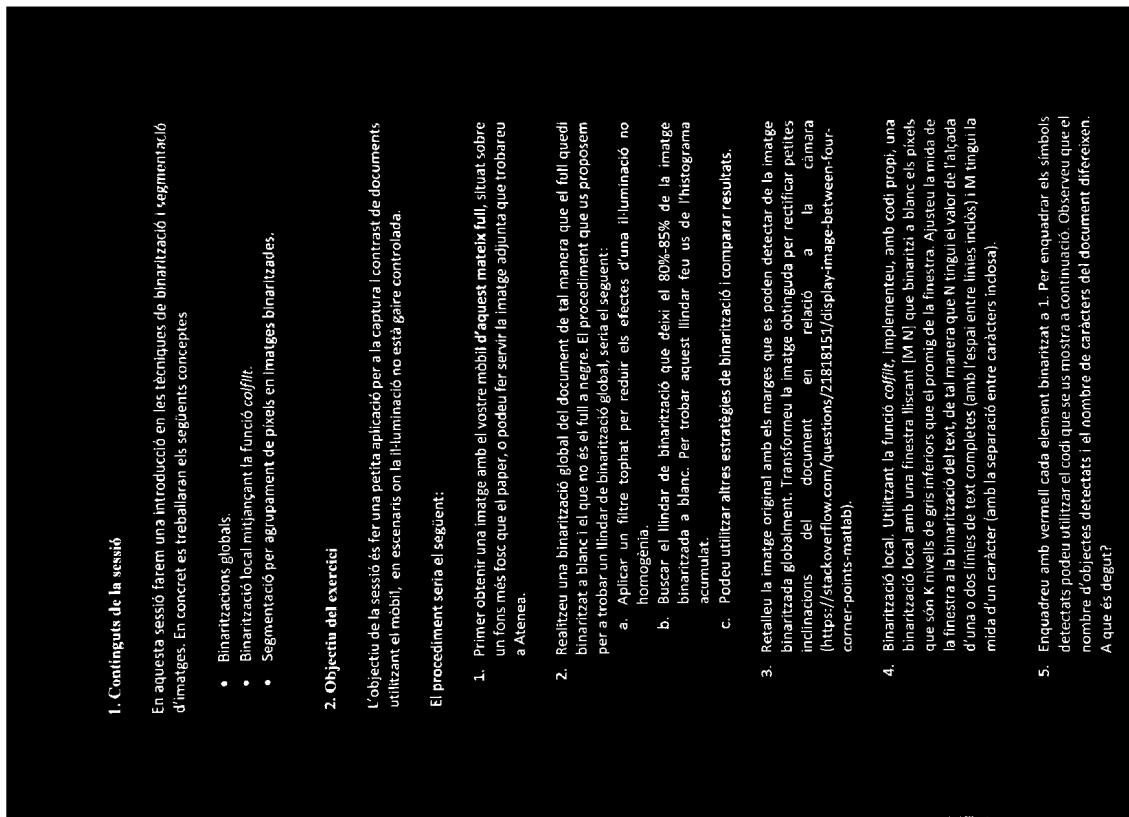
Output amb zoom:



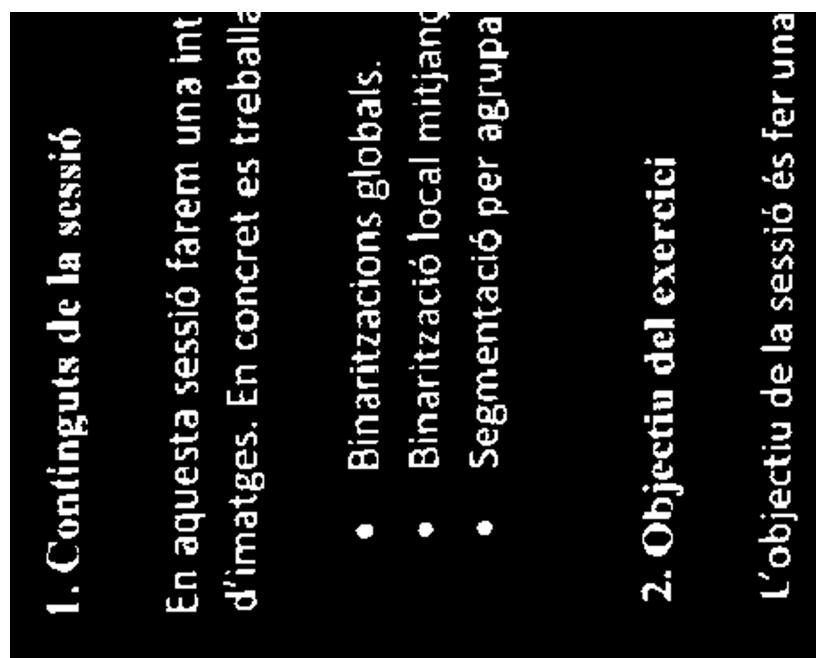
Amb $k = 60$ ens hem passat, ja que alguns caràcters es divideixen o són molt primis, impossibilitant el seu reconeixement.

El valor ideal k , per tant, seria $k = 50$.

Output:



Output amb zoom:



Codi binaritzacio:

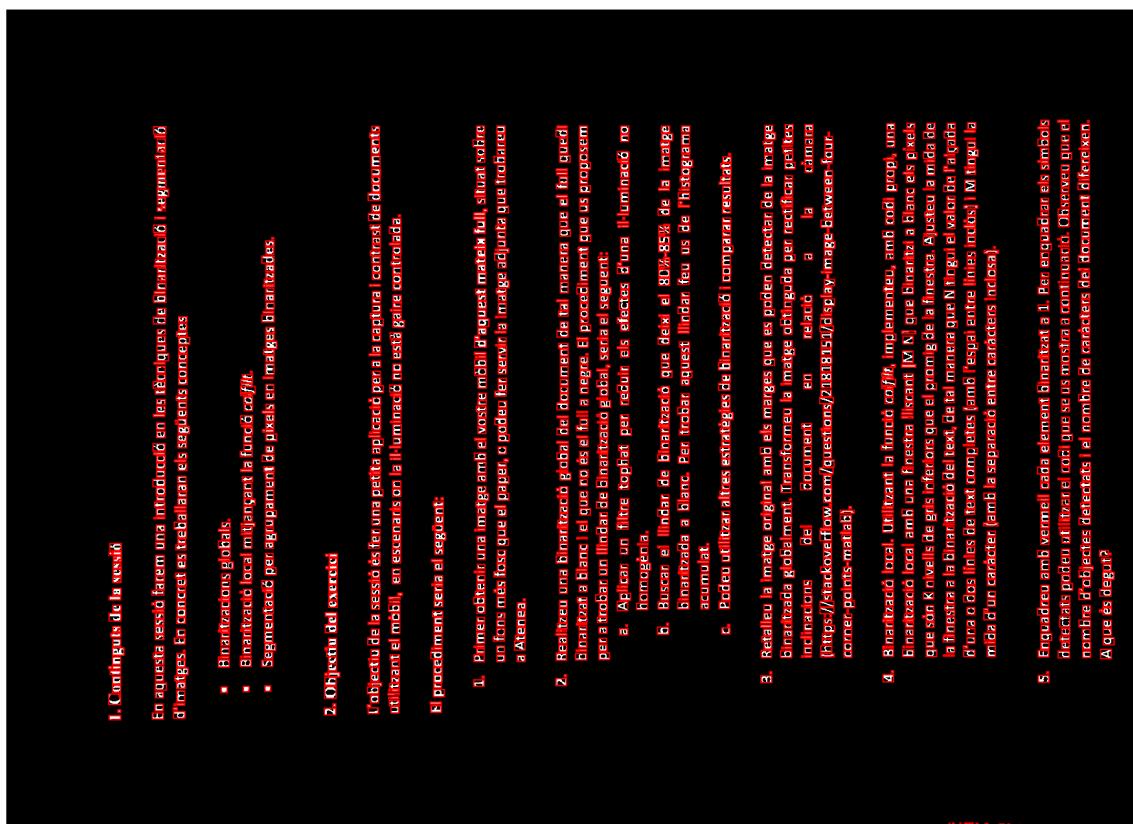
```
%binaritzacio prova 5  
binLocal5 = blkproc(imgTransformedfiltered, [1, 1], [25, 70], @myfun, 50);  
figure  
imshow(binLocal5);
```

- Pas 5: Encuadrament en vermell i nombre de components

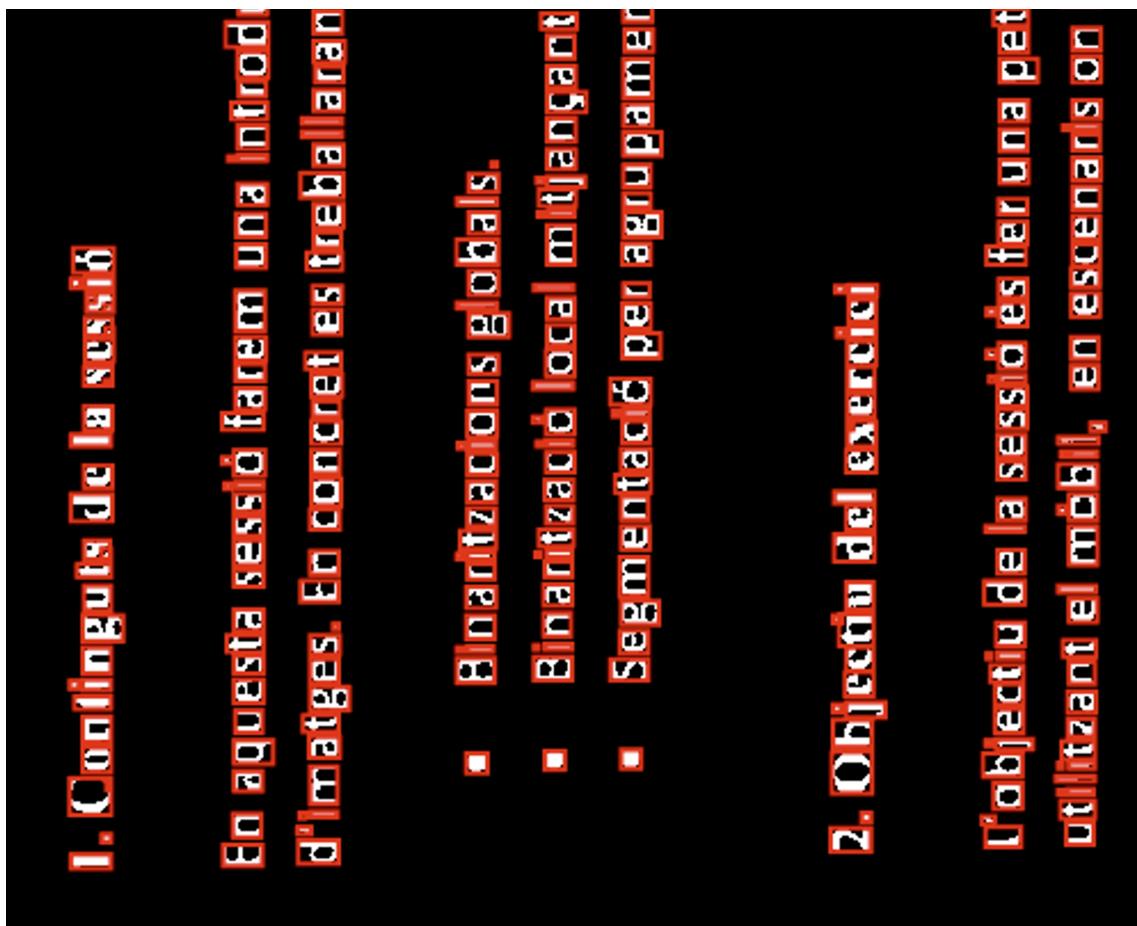
Un cop ja tenim binaritzada d'una forma bastant correcta la imatge, ja podem analitzar les components connexes d'aquesta. Per fer això, encuadrarem en vermell cada component, amb el següent codi:

```
%nombre components  
labeledImage = bwconncomp(binLocal5);  
labeledImage.NumObjects  
  
% el vermell  
measurements = regionprops(labeledImage, 'BoundingBox');  
for k = 1 : length(measurements)  
    bb = measurements(k).BoundingBox;  
    rectangle('Position', [bb(1),bb(2),bb(3),bb(4)], 'EdgeColor', 'red', 'LineWidth', 1 );  
end
```

Output:



Output amb zoom:



Observem que els resultats són força bons, ja que quasi tots els caràcters són una component connexa individual, i a més tenen una bona forma com perquè es puguin reconèixer posteriorment per algun algorisme de reconeixement de caràcters.

No obstant, cal dir que si observem el nombre components connexes que es reconeixen en total, aquest nombre és més gran que el nombre total de caràcters que realment hi ha al full:

ans =

2173

La raó per la qual aquest nombre és més alt principalment és perquè hi ha caràcters que són formats per més d'una component connexa. Això és degut a

que per exemple per el caràcter “i”, el pal es reconeix com una component diferent al punt, ja que no es tracta de pixels connectats entre si. El mateix passa amb molts altres caràcters, com aquelles lletres que tenen algun accent.

Finalment, el fet d'haver corregit la il·luminació anteriorment, ha produït que l'últim pas, **el pas 6** (eliminació de petites taques) no el considerem necessari, ja que el fons ens queda totalment negre.

Sí que és veritat que a l'inferior de la imatge queden unes petites taques fruit del retallat de la imatge:



El problema, és que tot i que aquestes taques són molt petites, també són molt petits els caràcters, de forma l'eliminació d'aquestes taques mantenint la mateixa qualitat de les components dels caràcters és molt difícil, ja que si per exemple realitzem un open i seguidament una reconstrucció d'aquest opening amb la imatge que teniem fins ara, hauriemp d'utilitzar un element estructurant molt petit, el qual no ens garanteix que s'eliminin aquestes taques.

De fet, hem intentat implementar això amb un element estructurant de mida 2, però els resultats, llavors de millorar, han empitjorat, ja que d'una banda

l'element estructurant és massa petit com per eliminar les taques, i d'altre és massa gran com per que es preservi la forma de cada caràcter. Codi:

```
se = strel('disk',2);
T = imopen(binLocal5, se);
binLocal5 = imreconstruct(T, binLocal5);
figure
imshow(binLocal5)
```

Per tant és millor no realitzar això.

Finalment llavors ja tenim la imatge final:

1. Continguts de la sessió

En aquesta sessió farem una introducció en les tècniques de binarització i segmentació d'imatges. En concret es treballaran els següents conceptes

- Binaritzacions globals.
- Binarització local mitjançant la funció `coffit`.
- Segmentació per agrupament de pixels en imatges binaritzades.

2. Objectiu del exercici

L'objectiu de la sessió és fer una petita aplicació per a la captura i contrast de documents utilitzant el mòbil, en escenaris on la lluminació no està gaire controlada.

El procediment seria el següent:

1. Primer obtenir una imatge amb el vostre mòbil d'aquest mateix full, situat sobre un fons més fosc que el paper, o podeu fer servir la imatge adjunta que trobareu a Atenea.
2. Realitzeu una binarització global del document de tal manera que el full quedí binaritzat a blanc i el que no és el full a negre. El procediment que us proposem per a trobar un líndar de binarització global, seria el següent:
 - a. Aplicar un filtre top-hat per reduir els efectes d'una il·luminació no homogènia.
 - b. Buscar el líndar de binarització que deixi el 80%-85% de la imatge binaritzada a blanc. Per trobar aquest líndar feu us de l'histograma acumulat.
 - c. Podeu utilitzar altres estratègies de binarització i comparar resultats.
3. Retalleu la imatge original amb els marges que es poden detectar de la imatge binaritzada. Globalment. Transformeu la imatge obtinguda per rectificar petites inclinacions del document en relació a la càmera (<https://stackoverflow.com/questions/21818151/display-image-between-four-corner-points-matlab>).
4. Binarització local. Utilitzant la funció `coffit`, implementeu, amb codi propi, una binarització local amb una finestra lliscant $[M \times N]$ que binaritzi a blanc els pixels que són K nivells de gris inferiors que el promig de la finestra. Adjsteu la mida de la finestra a la binarització del text, de tal manera que tingui el valor de l'alçada d'una o dos línies de text completes (amb l'estai entre línies inclos). I M tingui la mida d'un caràcter (amb la separació entre caràcters inclosa).
5. Enquadreu amb vermell cada element binaritzat a 1. Per enquadrar els símbols detectats podeu utilitzar el codi que seu mostra a continuació. Observeu que el nombre d'objectes detectats i el nombre de caràcters del document difereixen. A què es deurat?