Experimentos

Autores(Grupo - D):

1. Alejandro, Kenny
2. Gorriz, Oriol
3. Hossain, Tanvir

ÍNDICE

Fase Inicial

Objetivos

- Encontrar los umbrales de comparación de histogramas con los que se obtiene mejores resultados.
- Encontrar el numero de histogramas de modelo con el que ha ser similar el de una ventana, para ser considerado como buena muestra.
- Deteminar cuando cladificar un patch como bueno apartir de los indicadores globales de numero de ventanas buenas que hay para cada tamaño de ventana.

Recursos

- **Patch manual**
- Histogramas de color RB (derivados del modelo RGB)
- Histogramas de color con bins 8 y 16 por componente

8bins

```
isRGB = true;
bins = 8;
isAutomatic = false;
[infoMatrix,aditional_data] =TestWithConsole(isRGB,bins,isAutomatic);
```

16 bins

```
isRGB = true;
bins = 16;
isAutomatic = false;
[infoMatrix,aditional_data] =TestWithConsole(isRGB,bins,isAutomatic);
```

Fase Intermedia

Objetivos

- Verificar que el patch automatico es suficientemente bueno.
- Encontrar los umbrales y valores de la fase inicial para histogramas de color HS (dervidao de HSV)

Recursos

- **Patch automatico**
- Histogramas de color RB y HS (derivados del modelo RGB y HSV, respectivamente)
- Histogramas de color con bins 8 y 16 por componente

Patch manual vs automatico

Imagen del Barcelona

Usaremos las 10 primeras images de barcelona

Manual

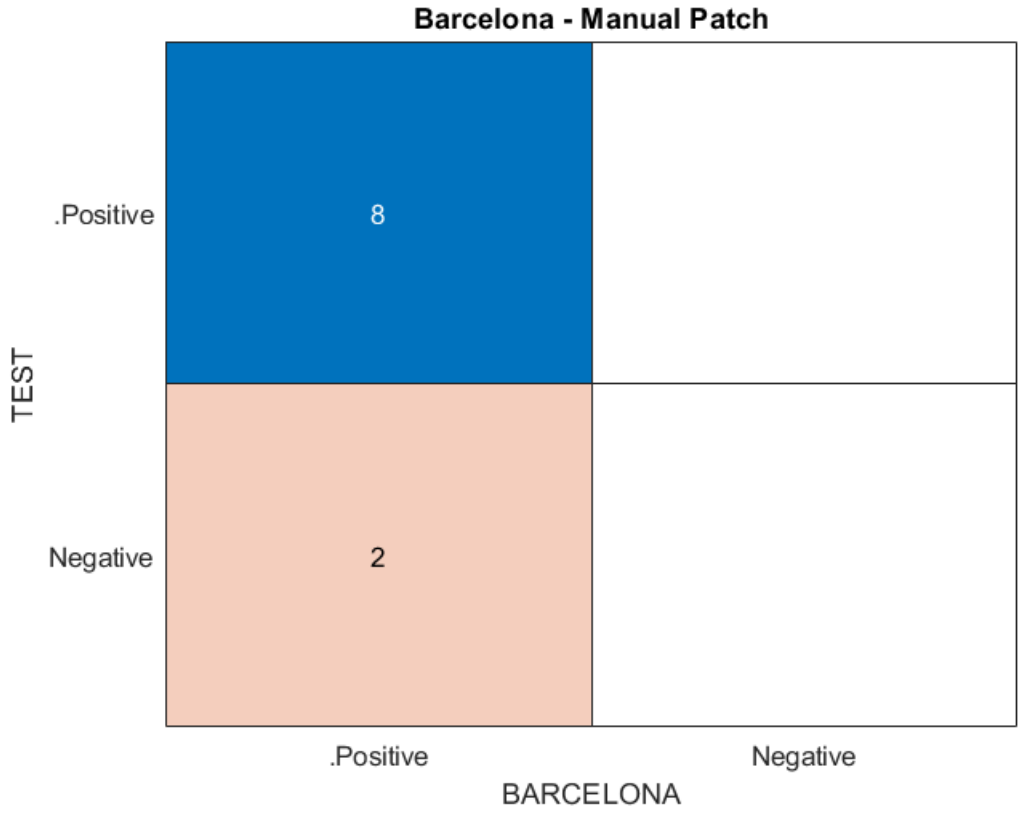
```
isAutomatic = false;
bins = 16;
isRGB = true;
[infoMatrix_manual,~] =TestWithConsole(isRGB,bins,isAutomatic);
```

ans = 10×3 table

	Club	Numbers	isBarcelona
1	"barcelona"	1	1
2	"barcelona"	2	1
3	"barcelona"	3	1
4	"barcelona"	4	1
5	"barcelona"	5	1
6	"barcelona"	6	0
7	"barcelona"	7	1
8	"barcelona"	8	1
9	"barcelona"	9	0
10	"barcelona"	10	1

```
figure
```

```
confusion_table_manual = buildConfusionTable(infoMatrix_manual,"barcelona");
drawConfusionTable(confusion_table_manual);
title("Barcelona - Manual Patch");
```



Automatico

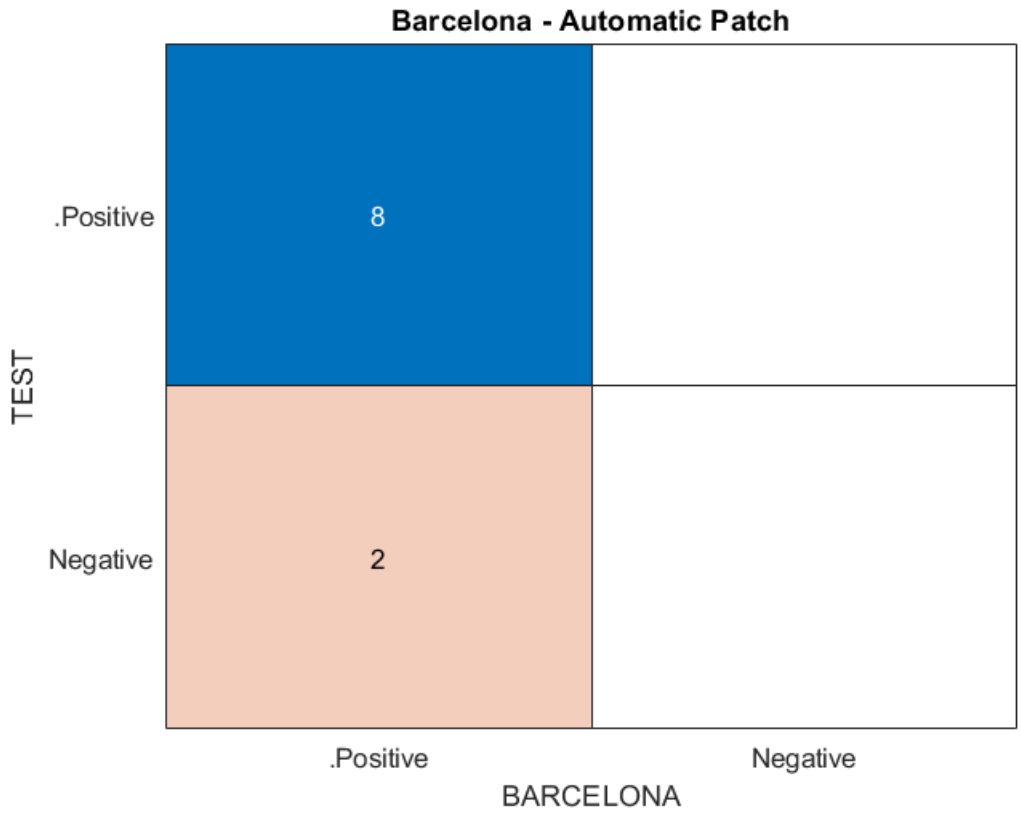
```
isAutomatic = true;
bins = 16;
isRGB = true;
[infoMatrix_automatico,~] =TestWithConsole(isRGB,bins,isAutomatic);
```

ans = 10×3 table

	Club	Numbers	isBarcelona
1	"barcelona"	1	1
2	"barcelona"	2	1
3	"barcelona"	3	1
4	"barcelona"	4	1
5	"barcelona"	5	1
6	"barcelona"	6	1
7	"barcelona"	7	0
8	"barcelona"	8	1

	Club	Numbers	isBarcelona
9	"barcelona"	9	0
10	"barcelona"	10	1

```
figure
confusion_table_automatico = buildConfusionTable(infoMatrix_automatico,"barcelona");
drawConfusionTable(confusion_table_automatico);
title("Barcelona - Automatic Patch");
```



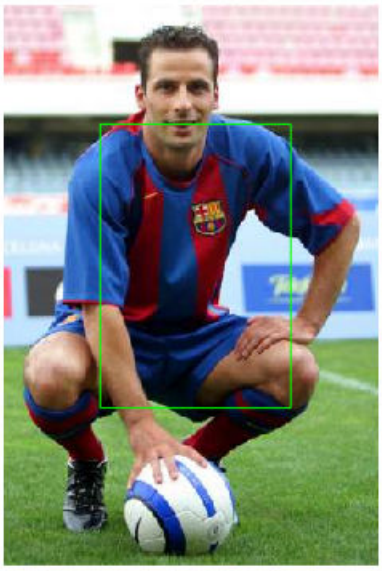

```
save("TestPatchBarcelona.mat","confusion_table_manual","confusion_table_automatico");
```

Path selecionado de las imagenes con fallos 6, 7, 9

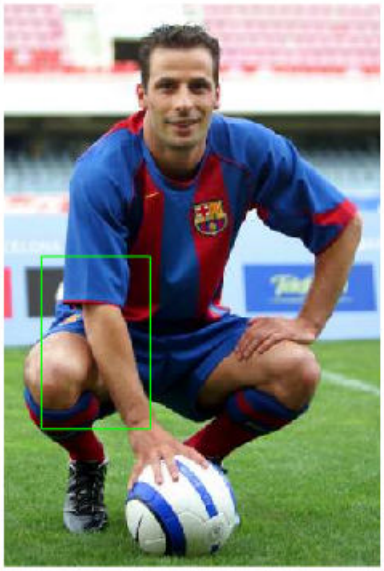
```
figure
clubName = "barcelona";
for i = [6,7,9]
    if(i<10)
        i_tmp = sprintf('%s%d', "0", i);
    else
        i_tmp = sprintf('%d', i);
    end
    path = sprintf('./imatges_equips/%s/%s.jpg',clubName,i_tmp);
    I = imread(path);
    figure
```

```
    subplot(1,2,1)
    imshow(I);
    title(sprintf("%d - Manual",i))
    rect = uint16(getrect);
    hold on
    rectangle('Position',rect,'EdgeColor','g');
    hold off
    subplot(1,2,2)
    imshow(I);
    title(sprintf("%d - Automatic",i))
    rect = bbBarca(I);
    hold on
    rectangle('Position',rect,'EdgeColor','g');
    hold off
end
```

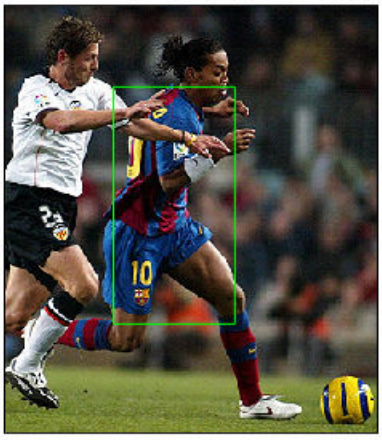
7 - Manual



7 - Automatic



9



9 - Automatic

Imagen de otros clubs

(acmilan, juventus,etc);

Usaremos 10 images selecionadas de forma aleatoria

```
teams = ["acmilan", "chelsea", "juventus", "liverpool", "madrid", "psv"];

clubName = (teams(1+round(rand(10,1)'*(size(teams,2)-1))))';
number = (1+round(rand(10,1)'*35))';
selectedImages = sortrows(table(clubName,number),"clubName");
```

selectedImages = 10×2 table

	clubName	number
1	"acmilan"	24
2	"chelsea"	18
3	"chelsea"	14
4	"juventus"	5
5	"juventus"	22
6	"liverpool"	14
7	"liverpool"	16
8	"madrid"	8
9	"madrid"	21
10	"psv"	9

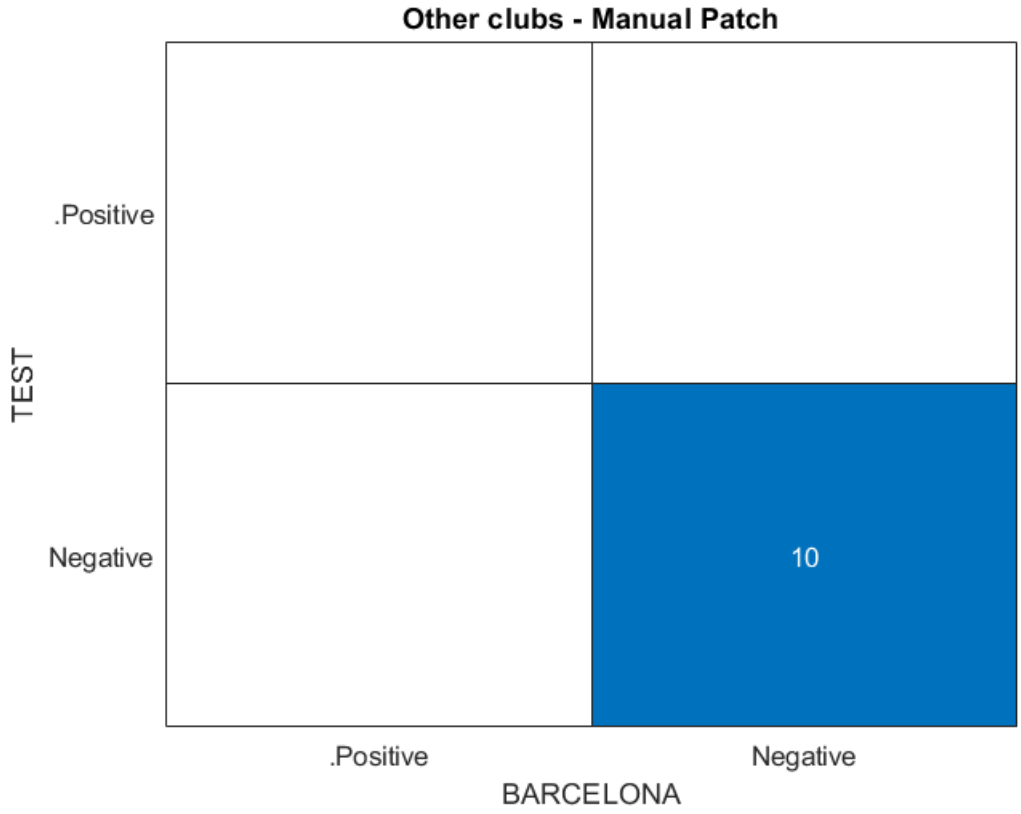
Manual

```
isAutomatic = false;
bins = 16;
isRGB = true;
[infoMatrix_manual,~] = TestWithConsole(isRGB,bins,isAutomatic);
```

ans = 10×3 table

	Club	Numbers	isBarcelona
1	"acmilan"	24	0
2	"chelsea"	18	0
3	"chelsea"	14	0
4	"juventus"	5	0
5	"juventus"	22	0
6	"liverpool"	14	0
7	"liverpool"	16	0
8	"madrid"	8	0
9	"madrid"	21	0
10	"psv"	9	0

```
figure
confusion_table_manual = buildConfusionTable(infoMatrix_manual,"barcelona");
drawConfusionTable(confusion_table_manual);
title("Other clubs - Manual Patch");
```



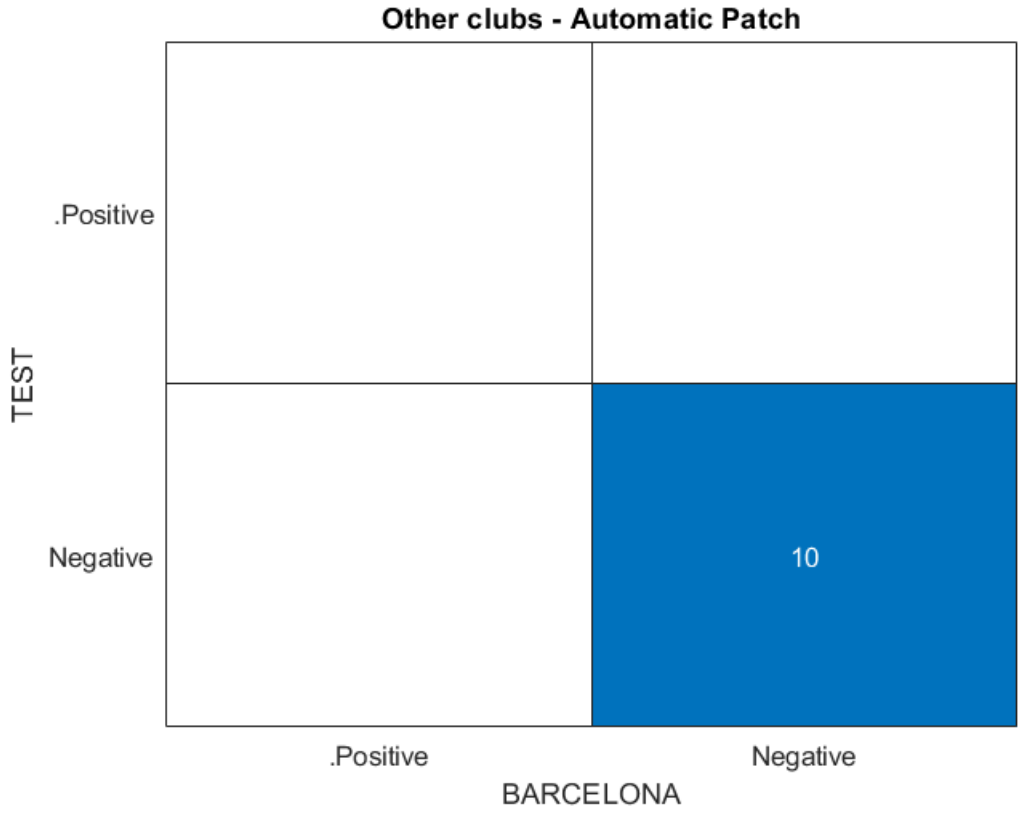
Automatico

```
isAutomatic = true;
bins = 16;
isRGB = true;
[infoMatrix_automatic,~] = TestWithConsole(isRGB,bins,isAutomatic);
```

```
ans = 10×3 table
```

	Club	Numbers	isBarcelona
1	"acmilan"	24	0
2	"chelsea"	18	0
3	"chelsea"	14	0
4	"juventus"	5	0
5	"juventus"	22	0
6	"liverpool"	14	0
7	"liverpool"	16	0
8	"madrid"	8	0
9	"madrid"	21	0
10	"psv"	9	0

```
figure
confusion_table_manual = buildConfusionTable(infoMatrix_manual,"barcelona");
drawConfusionTable(confusion_table_manual);
title("Other clubs - Automatic Patch");
```




```
save("TestPatchOtherClubs.mat","confusion_table_manual","confusion_table_automatico");
```

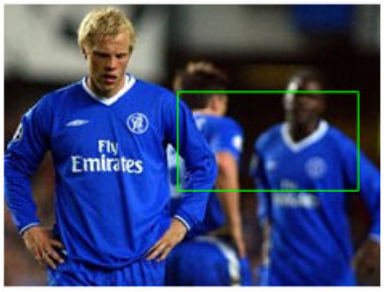
```
figure
listclubs= selectedImages.clubName';
listNumber = selectedImages.number'
for it = 1:length(listNumber)
    i = listNumber(it);
    club = listclubs(it);
    if(i<10)
        i_tmp = sprintf('%s%d', "0", i);
    else
        i_tmp = sprintf('%d', i);
    end
    path = sprintf('./imatges_equips/%s/%s.jpg',club,i_tmp);
    I = imread(path);
    figure
    subplot(1,2,1)
    imshow(I);
    title(sprintf("%d - Manual",i))
    rect = uint16(getrect);
    hold on
    rectangle('Position',rect,'EdgeColor','g');
    hold off
    subplot(1,2,2)
    imshow(I);
    title(sprintf("%d - Automatic",i))
    g_box = bbBarca(I);
    %tshirt_box gives start point( "x,y" => "col,row")and the size oo the box (width and height
    % tshirt_box = determineShirt();
    if ~isempty(g_box)
        gbox = sortrows(g_box,[3 4])
        % take the maximum bounding box
        rect = gbox(1,:);
    else
        %Doing a centered crop of the image
        pct = 80/100; % A percentage of each the size
        rect = floor([size(I,2)*(1-pct)*0.5 size(I,1)*(1-pct)*0.5 size(I,2)*pct size(I,1)*pct])
    end
    hold on
    rectangle('Position',rect,'EdgeColor','g');
    hold off
end
```

24 - Manual

24 - Automatic

18 - Manual

18 - Automatic

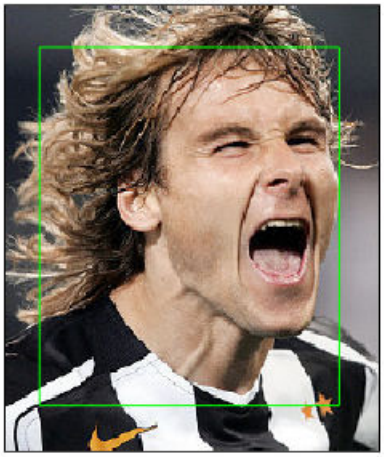


14 - Manual

14 - Automatic

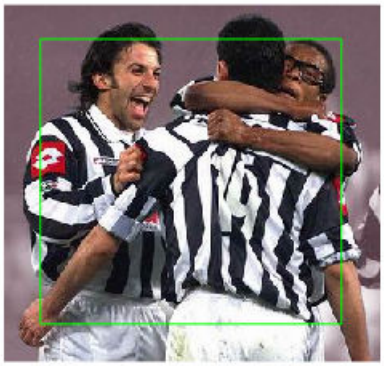
5 - Manual

5 - Automatic

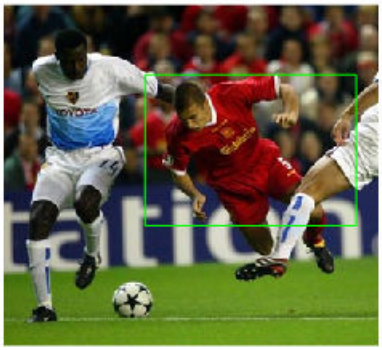


22 - Manual

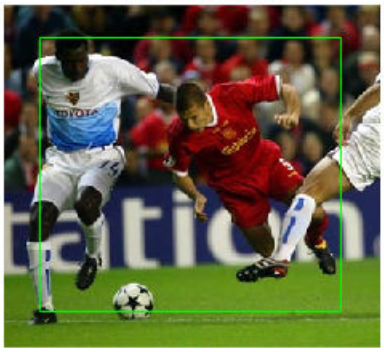
22 - Automatic



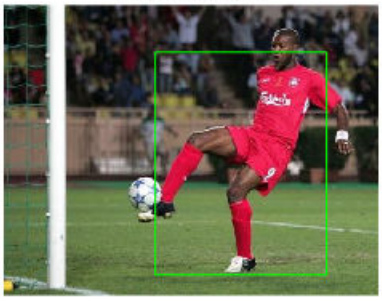
14 - Manual



14 - Automatic



16 - Manual

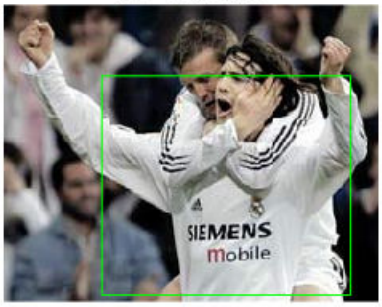


16 - Automatic

8 - Manual

8 - Automatic

21 - Manual



21 - Automatic

9 - Manual

9 - Automatic

Parametros de la fase inicial para HSV

8bins

```
isRGB = false;
bins = 8;
TestWithConsole(isRGB,bins);
```

16 bins

```
isRGB = false;
bins = 8;
TestWithConsole(isRGB,bins);
```

Fase Final

Objetivos

- Determinar el numero de bins mas apropiado.
- Deteminar el modelos de color (RGB o HSV) que nos da histogramas de color (RB o HS) mas robustos (mayor tasa de acierto)

Recursos

- Patch automatico
- Histogramas de color RB y HS (derivados del modelo RGB y HSV, respectivamente)
- Histogramas de color con bins 8 y 16 por componente

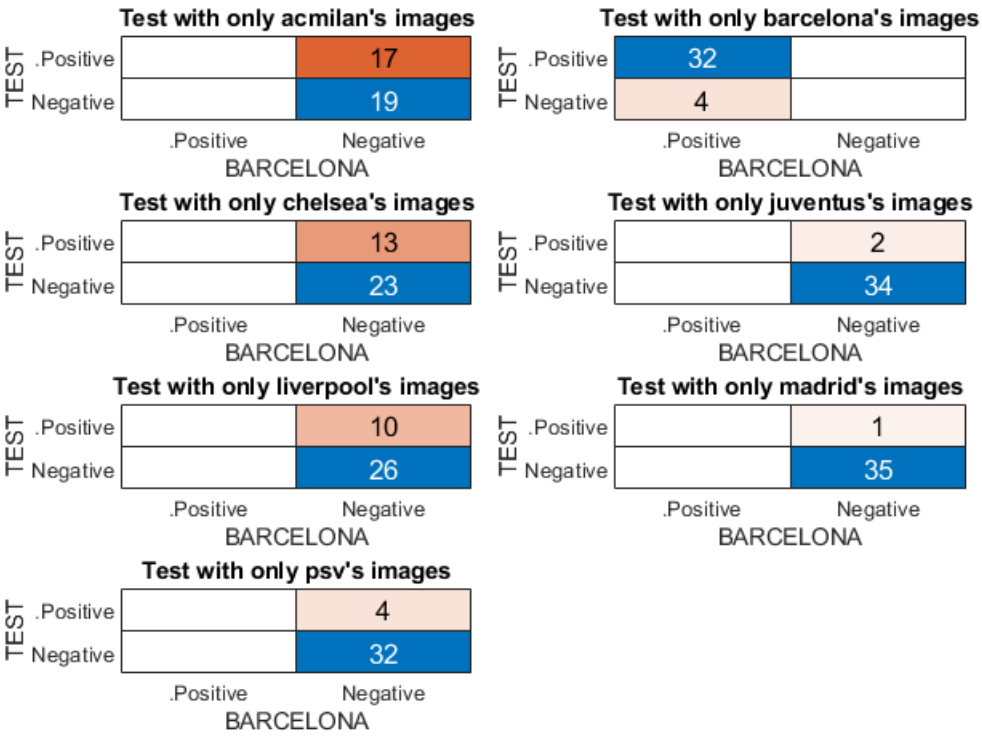
RGB

8bins

8 bins por componente

Test con imagenes de un mismo club

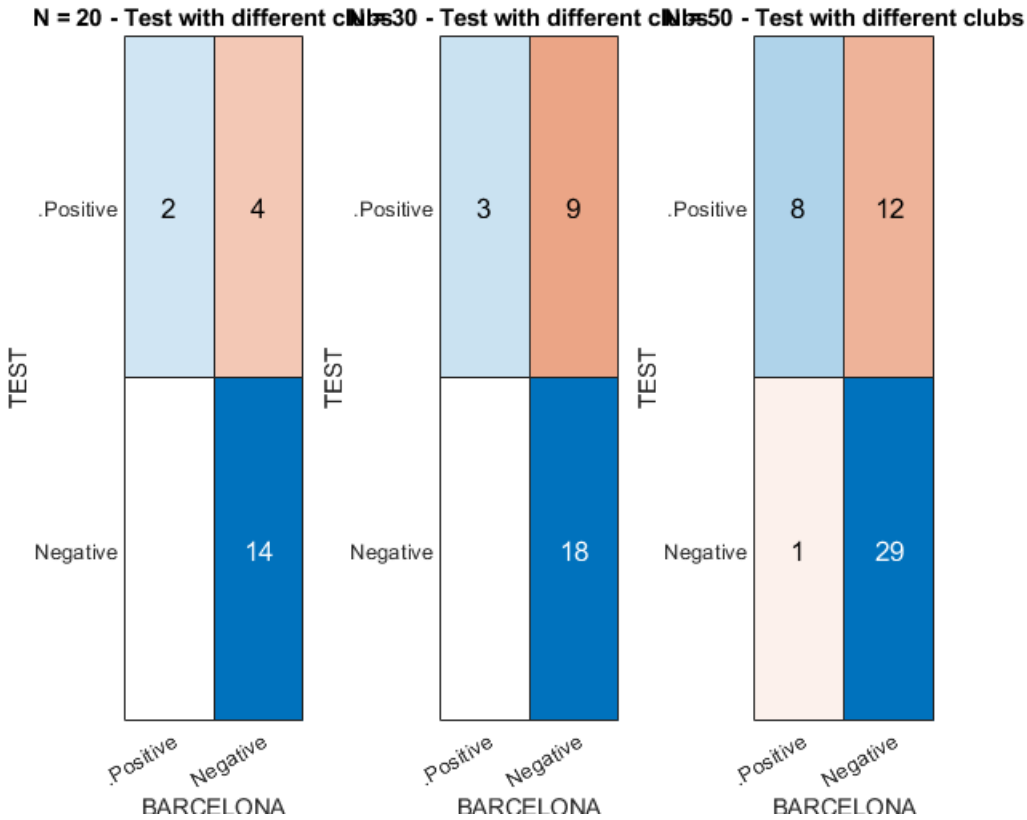
```
teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
isRandom = false;% Means it is not important the second paremeter
isRGB = true;% RGB
bins = 8;
n_clubs_images = 36;
figure
for i = 1:size(teams,2)
    imgInfo = [ repmat(teams(i),n_clubs_images,1) uint8([1:n_clubs_images]')];
    infoMatrix= testWithMultiplesClubs(isRandom,0,imgInfo,isRGB, bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = sprintf("Test with only %s's images", teams(i));
    confusion_table.n_samples = n_clubs_images;
    subplot(ceil(size(teams,2)/2),2,i)
    drawConfusionTable(confusion_table);
    title(confusion_table.comment);
    test_number= saveResults("TestsResults_RGB_8.mat",confusion_table);
end
```



Test con imagenes de varios clubs

N = 20, 30, 50

```
n_samples = [20,30,50];
figure
for i = 1:size(n_samples,2)
    isRGB = true;
    isRandom = true;
    bins = 8;
    infoMatrix = testWithMultiplesClubs(isRandom,n_samples(i),[],isRGB,bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = "Test with different clubs";
    confusion_table.n_samples = n_samples(i);
    subplot(1,size(n_samples,2),i)
    drawConfusionTable(confusion_table);
    text = sprintf('N = %d - %s',n_samples(i),confusion_table.comment)
    title(text);
    saveResults('TestsResults_RGB_8.mat',confusion_table)
end
```

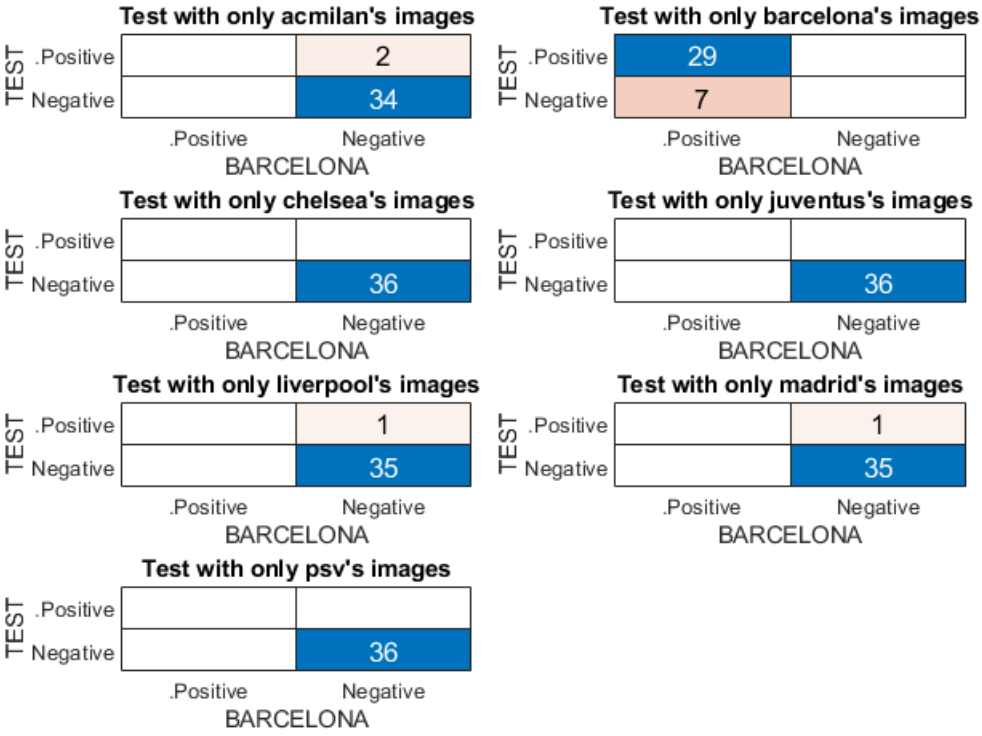


16 bins

16 bins por componente

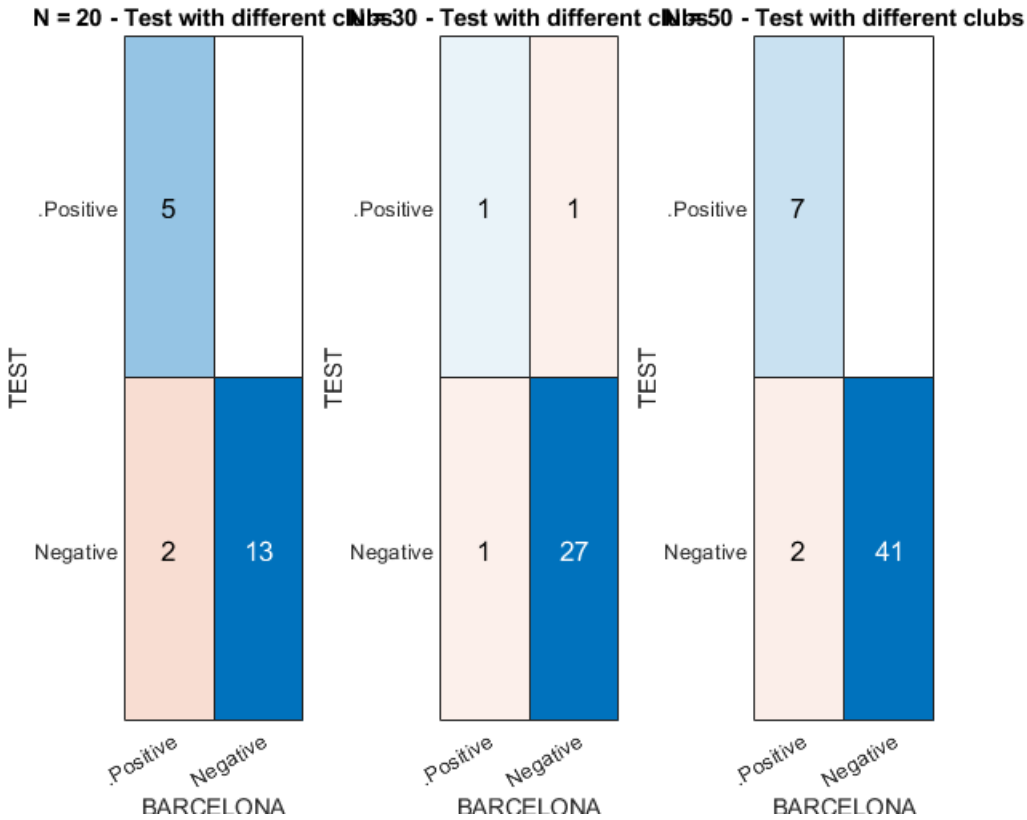
Test con imagenes de un mismo club

```
teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
isRandom = false;% Means it is not important the second paremeter
isRGB = true;% RGB
bins  = 16;
n_clubs_images = 36;
for i = 1:size(teams,2)
    imgInfo = [ repmat(teams(i),n_clubs_images,1) uint8([1:n_clubs_images]')];
    infoMatrix= testWithMultiplesClubs(isRandom,0,imgInfo,isRGB, bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = sprintf("Test with only %s's images", teams(i));
    confusion_table.n_samples = n_clubs_images;
    subplot(ceil(size(teams,2)/2),2,i)
    drawConfusionTable(confusion_table);
    title(confusion_table.comment);
    test_number= saveResults("TestsResults_RGB_16.mat",confusion_table)
end
```



Test con imagenes de varios clubs

```
n_samples = [20,30,50];
figure
for i = 1:size(n_samples,2)
    isRGB = true;
    isRandom = true;
    bins = 16;
    infoMatrix = testWithMultiplesClubs(isRandom,n_samples(i),[],isRGB,bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = "Test with different clubs";
    confusion_table.n_samples = n_samples(i);
    subplot(1,size(n_samples,2),i)
    drawConfusionTable(confusion_table);
    text = sprintf('N = %d - %s',n_samples(i),confusion_table.comment)
    title(text);
    saveResults('TestsResults_RGB_16.mat',confusion_table)
end
```

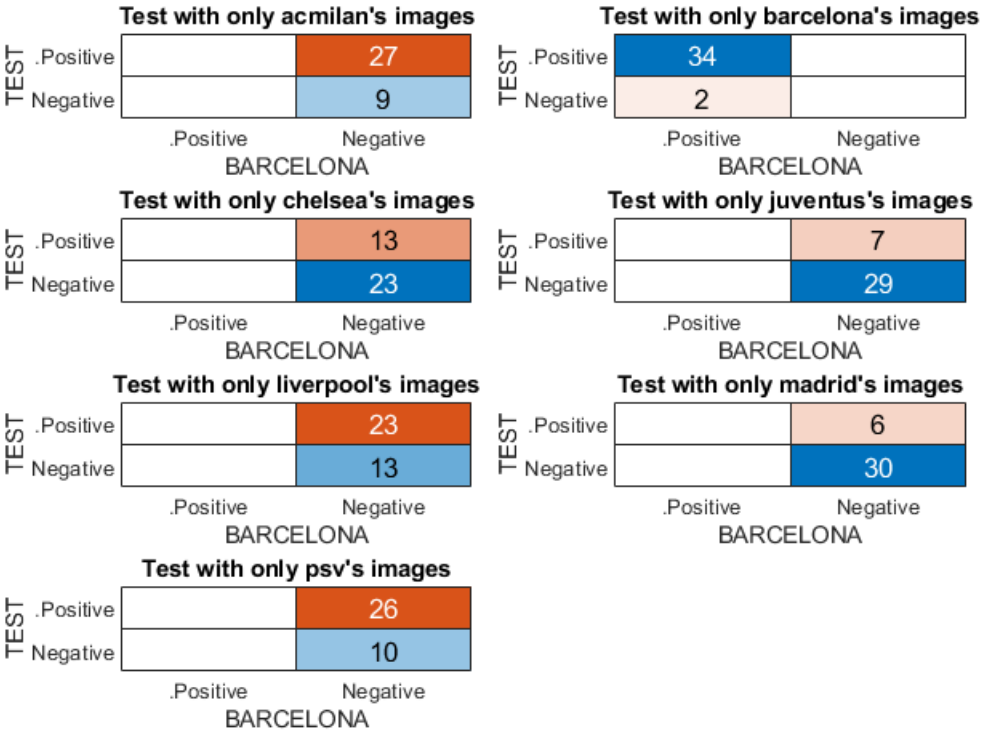


HSV

8bins

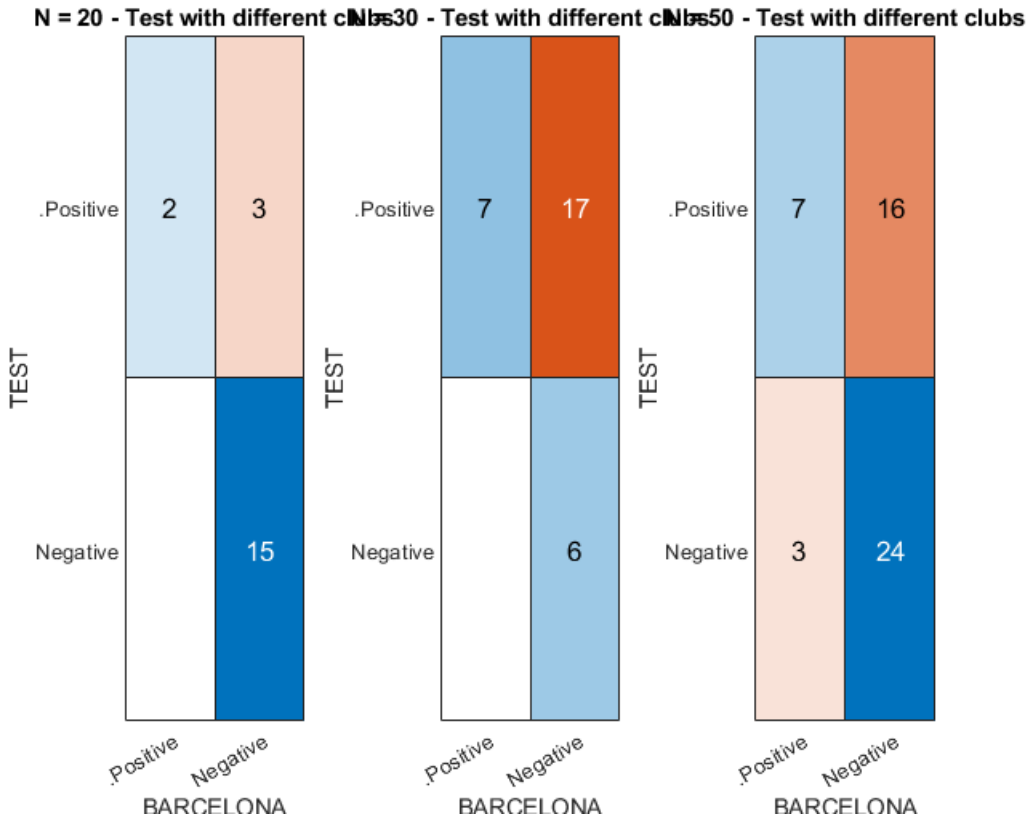
Test con imagenes de un mismo club

```
teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
isRandom = false;% Means it is not important the second paremeter
isRGB = false;% HSV
bins = 8;
n_clubs_images = 36;
figure
for i = 1:size(teams,2)
    imgInfo = [ repmat(teams(i),n_clubs_images,1) uint8([1:n_clubs_images]')];
    infoMatrix= testWithMultiplesClubs(isRandom,0,imgInfo,isRGB, bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = sprintf("Test with only %s's images", teams(i));
    confusion_table.n_samples = n_clubs_images;
    subplot(ceil(size(teams,2)/2),2,i)
    drawConfusionTable(confusion_table);
    title(confusion_table.comment);
    test_number= saveResults("TestsResults_HSV_8.mat",confusion_table);
end
```



Test con imagenes de varios clubs

```
n_samples = [20,30,50];
isRGB = false;
isRandom = true;
bins = 8;
figure
for i = 1:size(n_samples,2)
    infoMatrix = testWithMultiplesClubs(isRandom,n_samples(i),[],isRGB,bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = "Test with different clubs";
    confusion_table.n_samples = n_samples(i);
    subplot(1,size(n_samples,2),i)
    drawConfusionTable(confusion_table);
    text = sprintf('N = %d - %s',n_samples(i),confusion_table.comment);
    title(text);
    saveResults('TestsResults_HSV_8.mat',confusion_table)
end
```

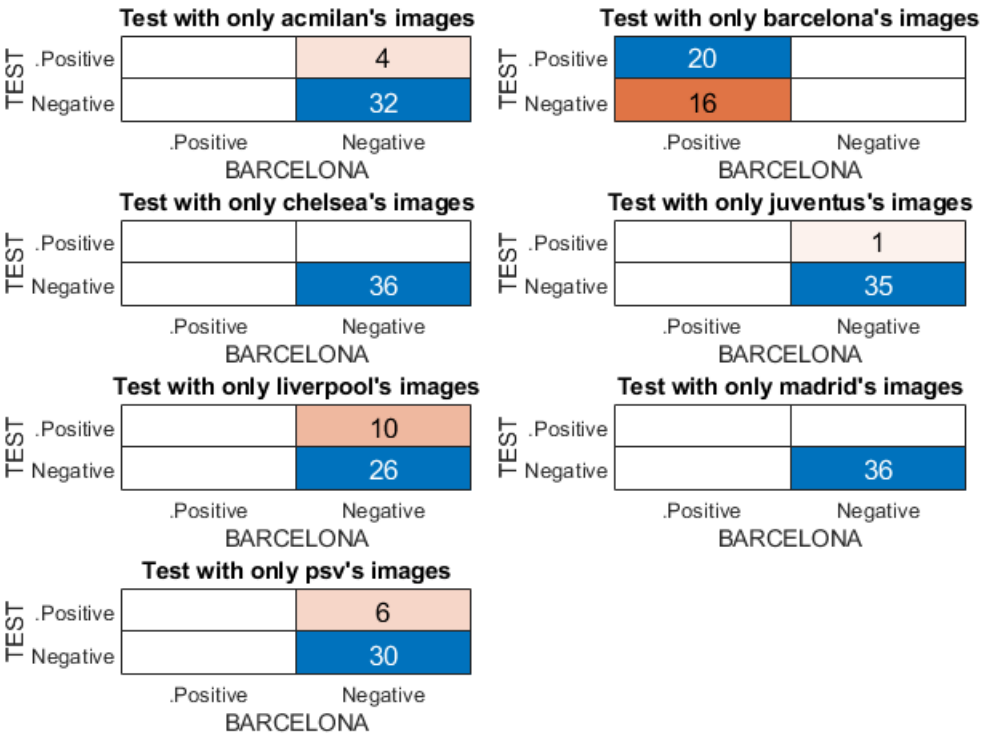


16 bins

16 por componente

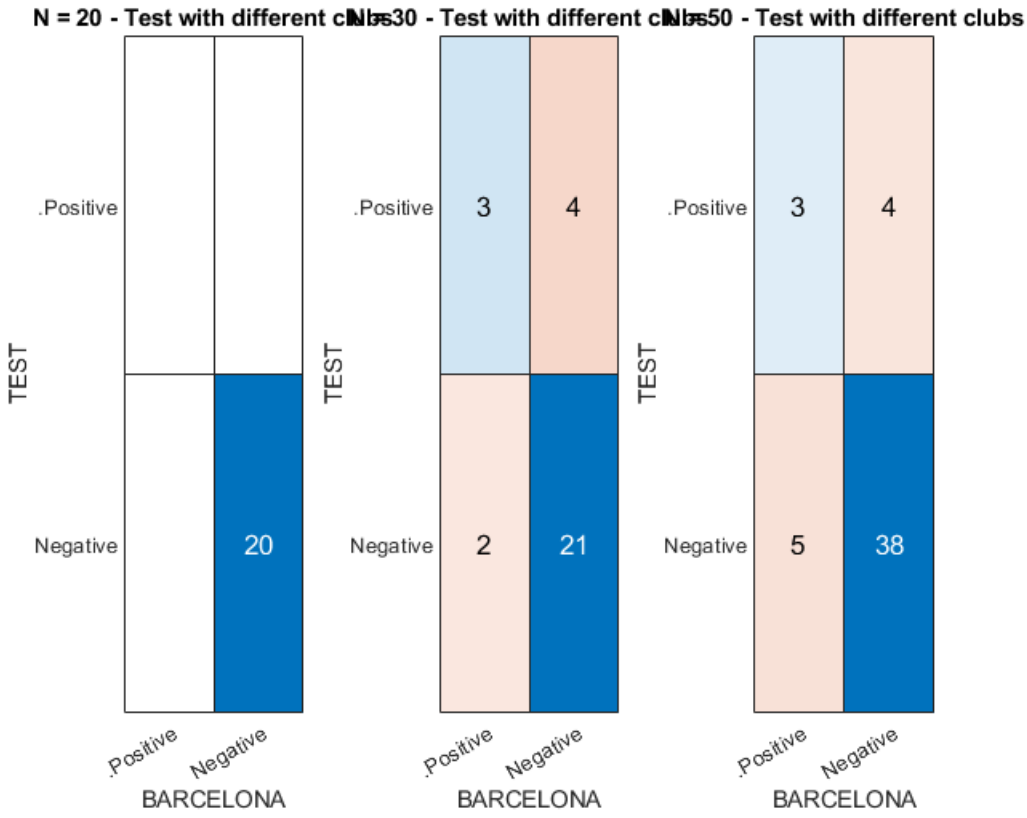
Test con imagenes de un mismo club

```
teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
isRandom = false;% Means it is not important the second paremeter
isRGB = false;% HSV
bins  = 16;
n_clubs_images = 36;
figure
for i = 1:size(teams,2)
    imgInfo = [ repmat(teams(i),n_clubs_images,1) uint8([1:n_clubs_images]')];
    infoMatrix= testWithMultiplesClubs(isRandom,0,imgInfo,isRGB, bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = sprintf("Test with only %s's images", teams(i));
    confusion_table.n_samples = n_clubs_images;
    subplot(ceil(size(teams,2)/2),2,i)
    drawConfusionTable(confusion_table);
    title(confusion_table.comment);
   test_number= saveResults("TestsResults_HSV_16.mat",confusion_table)
end
```



Test con imagenes de varios clubs

```
n_samples = [20,30,50];
bins = 16;
isRGB = false;
isRandom = true;
figure
for i = 1:size(n_samples,2)
    infoMatrix = testWithMultiplesClubs(isRandom,n_samples(i),[],isRGB,bins);
    confusion_table = buildConfusionTable(infoMatrix,"barcelona");
    confusion_table.comment = "Test with different clubs";
    confusion_table.n_samples = n_samples(i);
    subplot(1,size(n_samples,2),i)
    drawConfusionTable(confusion_table);
    text = sprintf('N = %d - %s',n_samples(i),confusion_table.comment)
    title(text);
    saveResults('TestsResults_HSV_16.mat',confusion_table)
end
```



Funciones

Funciones para test

```
function  [results] = testWithAClub(clubName,range,isAutomatic,isRGB,bins)
    %Input:
    % clubName : string
    % range : array
    % isAutomatic: boolean
    % bins: boolean
    % Output
    % results : Object with the anwers of is theris a Barcelona player and
    % additional information about that decision.
    teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
    if sum(teams == clubName) ~= 1
        error('There is no club with the given name:%s', clubName);
    end
    results.exists = [];
    results.additional_data = [];
    results.additional_data.good_sample = [];
    results.additional_data.bad_sample = [];
    for i = range
        if(i<10)
            i_tmp = sprintf('%s%d', "0", i);
        else
            i_tmp = sprintf('%d', i);
        end
```

```
        path = sprintf('./imatges_equips/%s/%s.jpg',clubName,i_tmp);
        I = imread(path);

        if isAutomatic
            g_box = bbBarca(I);
            %tshirt_box gives start point( "x,y" => "col,row")and the size oo the box (width ar
            % tshirt_box = determineShirt();
            if ~isempty(g_box)
                gbox = sortrows(g_box,[3 4]);
                % take the maximum bounding box
                tshirt_box = gbox(1,:);
            else
                %Doing a centered crop of the image
                pct = 80/100; % A percentage of each the size
                tshirt_box = floor([size(I,2)*(1-pct)*0.5 size(I,1)*(1-pct)*0.5 size(I,2)*pct s
            end
        else
%             figure
            imshow(I);
            tshirt_box = uint16(getrect);
        end
        I = imcrop(I,tshirt_box);
        if isRGB
            if bins == 8
                load histogramas_modelo_rgb_8.mat histModels
            else
               %bins == 16
                load histogramas_modelo_rgb_16.mat histModels
            end
            [exists,additional_data] = isThereABarcelonaPlayerRGB(I,histModels,bins); % RGB
        else
            if bins == 8
                load histogramas_modelo_hsv_8.mat histModels
            else
               %bins == 16
                load histogramas_modelo_hsv_16.mat histModels
            end
            [exists,additional_data] = isThereABarcelonaPlayerHSV(I,histModels,bins); % HSV
        end
        results.exists = [results.exists;exists];
        aux = [];
        aux = additional_data.good_sample;
        results.additional_data.good_sample = [results.additional_data.good_sample; aux];
        aux = [];
        aux = additional_data.bad_sample;
        results.additional_data.bad_sample  = [results.additional_data.bad_sample; aux];
    end

end

function infoMatrix =  testWithMultiplesClubs(isRandom,numberTests,imgInfo,isRGB,bins)
     %Input:
     % - isRandom indicates if the images are given by the user
     % as "imgInfo" or created randomly.
```

```
     % - "imgInfo" is an matrix two columns: "names" and "numbers". It is only considered when
     %   isRandom is false.
     % - numberTests specifys the number of tests. It is only considered when
     %   isRandom is true.
     % Output:
     % -infomatrix with three columns: nameclub numimage and isBarcelona

    teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
    n_equips = 7;
    n_img_test = numberTests;
    if ~isRandom
        n_img_test = size(imgInfo,1);
    end
    n_imgs_available = 36;
    Club = [];
    Number = [];
    isBarcelona = [];
    for i = 1:n_img_test
       if (isRandom)
           selectedTeam = teams(floor(1 + rand *(n_equips-1)));
           number = floor(1 + rand *(n_imgs_available-1));
       else
           selectedTeam = imgInfo(i,1);
            number = str2double(imgInfo(i,2));
       end
       Club = [Club ,selectedTeam];
       Number = [Number,number];
%        %-----------------------ATENTION ------------------------------
%        isAutomatic = false; %% Change ONCE THE BOX DETERMINER IS IMPLEMENTED
%       % -----------------------ATENTION ------------------------------
       isAutomatic = true;
       test_answer = testWithAClub(selectedTeam,[number],isAutomatic,isRGB,bins);
       isBarcelona = [isBarcelona , test_answer.exists];
    end
    infoMatrix(:,1) = Club;
    infoMatrix(:,2) = Number;
    infoMatrix(:,3) = isBarcelona;
end

function drawConfusionTable(confusion_table)
    m = [confusion_table.TP.counter, confusion_table.FP.counter;
         confusion_table.FN.counter, confusion_table.TN.counter];
    confusionchart(m,[".Positive","Negative"]');
    ylabel('TEST')
    xlabel('BARCELONA')
end

function [confusion_table] = buildConfusionTable(infoMatrix,referenceClub)
       %%Confusion Table  o contingencia
        %                              Barcelona
        %                    positivo             negativo
        %   T  positivo    truePositive  1   falsePositive  3
        %   E
        %   S  negativo    falseNegative 2   trueNegative   4
```

```
        %   T
        %     1 -> truepositve    TP
        %     2 -> falseNegative  FN
        %     3 -> falsePositive  FP
        %     4 -> trueNegative   TN
        confusion_table.TP.imgInfo = [];
        confusion_table.FN.imgInfo = [];
        confusion_table.FP.imgInfo = [];
        confusion_table.TN.imgInfo = [];
        confusion_table.TP.counter = 0;
        confusion_table.FN.counter = 0;
        confusion_table.FP.counter = 0;
        confusion_table.TN.counter = 0;
        for i = 1:size(infoMatrix,1)
           isPositive = logical(str2double(infoMatrix(i,3)));
           if (infoMatrix(i,1) == referenceClub)
               %Input: Barcelona
               %Desition:
               if isPositive
                 confusion_table.TP.counter = confusion_table.TP.counter + 1;%TP
                 confusion_table.TP.imgInfo = [confusion_table.TP.imgInfo;infoMatrix(i,1:2)];
               else
                 confusion_table.FN.counter  = confusion_table.FN.counter + 1;%FN
                 confusion_table.FN.imgInfo = [confusion_table.FN.imgInfo;infoMatrix(i,1:2)];
               end
           else
               %Input: No Barcelona
               %Desition:
               if isPositive
                 confusion_table.FP.counter  =  confusion_table.FP.counter + 1;%FP
                  confusion_table.FP.imgInfo = [confusion_table.FP.imgInfo;infoMatrix(i,1:2)];

               else
                  confusion_table.TN.counter  =  confusion_table.TN.counter + 1;%TN
                  confusion_table.TN.imgInfo = [confusion_table.TN.imgInfo;infoMatrix(i,1:2)];
               end
           end
        end
end

function test_number = saveResults(filename,new_confusion_table)
    try
        load(filename)
        max_num = length(cellfun('length',struct2cell(TestsResults)));
    catch ME
        TestsResults = [];
        max_num = 0;
    end
    test_number = max_num+1;
    custom_name_test = sprintf('TestResult_%d',test_number);
    TestsResults.(custom_name_test) = new_confusion_table;
    save(filename,"TestsResults")
end
```

```
function [infoMatrix, additional_data] = TestWithConsole(isRGB,bins,isAutomatic)
    teams = ["acmilan", "barcelona", "chelsea", "juventus", "liverpool", "madrid", "psv"];
    exit = false;
    first = true;
    Club = [];
    Numbers = [];
    isBarcelona = [];
    additional_data.good_sample = [];
    additional_data.bad_sample = [];
    while ~exit
       [exit, selectedTeam] = showconsole(first);
       if ~exit
            first =  false;
            prompt = 'Introduce a valid array of number between 1 and 36\nExample: [1 2 20 10]
            range = input(prompt);
            results = testWithAClub(teams(selectedTeam),range,isAutomatic,isRGB,bins);
            aux = [];
            aux = results.additional_data.good_sample;
            additional_data.good_sample = [additional_data.good_sample; aux];
            aux = [];
            aux = results.additional_data.bad_sample;
            additional_data.bad_sample  = [additional_data.bad_sample; aux];
            Club = [Club, repmat(teams(selectedTeam),1,length(range))];
            Numbers = [Numbers, range];
            isBarcelona = [isBarcelona, logical(results.exists)'];
            table(Club',Numbers',isBarcelona','VariableNames',{'Club','Numbers','isBarcelona'})
       end
    end
    infoMatrix(:,1) = Club;
    infoMatrix(:,2) = Numbers;
    infoMatrix(:,3) = isBarcelona;
end

function [exit, selectedTeam] = showconsole(first)
    exit =  false;
    madecision = false;
    selectedTeam = [];
    while  ~madecision
        if first
            first = false;
            str = 's';
            added_string = '[e] Exit\n';
        else
            message = [];
            message = ['Choose an option\n'...
                '[s] Select Team\n'...
                '[e] Exit\n'];
            str = input(message,'s');
            added_string = '[b] Back\n';
        end
        madecision = true;
```

```
        switch str
            case 's'
                isSelected = false;
                alert = [];
                while ~isSelected
                     prompt = [];
                     prompt  = [alert ,'Select a soccer club\n' ...
                         '[1] acmilan\n' ...
                         '[2] barcelona\n' ...
                         '[3] chelsea\n' ...
                         '[4] juventus\n' ...
                         '[5] liverpool\n' ...
                         '[6] madrid\n' ...
                         '[7] psv\n',added_string
                         ];
                     aux = input(prompt,'s');
                     isSelected = true;
                     if ~isempty(aux) && size(aux,1) == 1 && size(aux,2) == 1
                         if ( aux =='b' || aux == 'e')
                            if added_string == "[e] Exit\n" && aux =='e'
                              exit = true;
                            else
                                madecision = false;
                            end
                         elseif  aux >= '1' && aux <= '7'
                             selectedTeam = uint8(str2double(aux));
                         else
                              isSelected = false;
                         end
                     else
                          isSelected = false;
                     end
                     if ~isSelected
                            alert = 'Invalid option!!!\n\n';
                     end
                end
            case 'e'
                exit = true;
            otherwise
                madecision = false;
        end
    end
end
```

Funciones principales

```
function [existsABarcelonaPlayer, additional_data] =  isThereABarcelonaPlayerRGB(I,histModels,b
    [height,width,~] = size(I);
    reference_size = min(width,height);
    ratio = 0.9:-0.1:0.1;
    n_good_sample = zeros(1,length(ratio));
    n_bad_sample  = zeros(1,length(ratio));
    size_matrix =[];
    existsABarcelonaPlayer = false;
```

```
    min_size_window = 10;
    for k = 1:length(ratio)
       % Test with diferent window sizes
        window_size = floor(reference_size* ratio(k));
        if window_size >= min_size_window
            shift_ratio = 1/4;
            n_col = 1/shift_ratio* floor(width/window_size) - (1/shift_ratio - 1);
            n_row = 1/shift_ratio* floor(height/window_size) - (1/shift_ratio -1);
            for i = 1:n_row
                for j = 1:n_col
                    % Getting the squared window
                    ini_x = (j-1)*floor(window_size*shift_ratio);
                    ini_y = (i-1)*floor(window_size*shift_ratio);
                    window_crop = [ini_x ini_y  window_size window_size];
                    ITest = imcrop(I,window_crop);
                    %Getting the color histogram in RB

                    histTest = getRBColorHistogram(ITest,bins);
                    histTest = imgaussfilt(histTest);
                    %testing
                    n_test = 15;
                    % Compare and decide if it is a good or bad sample
                    is_similar = 0;
                    for q = 1:n_test
                        histoModel = histModels(:,:,q);
                        is_similar = is_similar + uint8(histSimilar(histoModel,histTest,bins));
                    end
                    if is_similar >= 4
                       %At leat there is histograms
                       n_good_sample(k) = n_good_sample(k) + 1 ;
                    else
                       n_bad_sample(k) = n_bad_sample(k) + 1 ;
                    end
                end
             end
        end
    end
    % Decide considereing the number of good and bad samples if there is a Barcelona's player o
    % the image
    if sum((n_good_sample./(n_good_sample + n_bad_sample))*100 >= 45) >= 1 || sum(n_good_sample
        existsABarcelonaPlayer = true;
    end
    additional_data.good_sample = n_good_sample;
    additional_data.bad_sample = n_bad_sample;
end

function [existsABarcelonaPlayer, additional_data] =  isThereABarcelonaPlayerHSV(I,histModels,b
    [height,width,~] = size(I);
    reference_size = min(width,height);
    ratio = 0.9:-0.1:0.1;
    n_good_sample = zeros(1,length(ratio));
    n_bad_sample  = zeros(1,length(ratio));
    size_matrix =[];
    existsABarcelonaPlayer = false;
```

```
    min_size_window = 10;
    for k = 1:length(ratio)
       % Test with diferent window sizes
        window_size = floor(reference_size* ratio(k));
        if window_size >= min_size_window
            shift_ratio = 1/4;
            n_col = 1/shift_ratio* floor(width/window_size) - (1/shift_ratio - 1);
            n_row = 1/shift_ratio* floor(height/window_size) - (1/shift_ratio -1);
            for i = 1:n_row
                for j = 1:n_col
                    % Getting the squared window
                    ini_x = (j-1)*floor(window_size*shift_ratio);
                    ini_y = (i-1)*floor(window_size*shift_ratio);
                    window_crop = [ini_x ini_y  window_size window_size];
                    ITest = imcrop(I,window_crop);
                    %Getting the color histogram in HSV
                    ITest = rgb2hsv(ITest);
                    histTest = getHSColorHistogram(ITest,bins);
                    histTest = imgaussfilt(histTest);
                    %testing
                    n_test = 15;
                    % Compare and decide if it is a good or bad sample
                    is_similar = 0;
                    for q = 1:n_test
                        histoModel = histModels(:,:,q);
                        is_similar = is_similar + uint8(histSimilar(histoModel,histTest,bins));
                    end
                    if is_similar >= 2
                       %At leat there is histograms
                       n_good_sample(k) = n_good_sample(k) + 1 ;
                    else
                       n_bad_sample(k) = n_bad_sample(k) + 1 ;
                    end
                end
             end
        end
    end
    % Decide considereing the number of good and bad samples if there is a Barcelona's player o
    % the image
    if sum((n_good_sample./(n_good_sample + n_bad_sample))*100 >= 45) >= 1 || sum (n_good_sampl
%   sum((n_good_sample./(n_good_sample + n_bad_sample))*100 >= 60) >= 2  %|| sum(n_good_sample >
        existsABarcelonaPlayer = true;
    end
    additional_data.good_sample = n_good_sample;
    additional_data.bad_sample = n_bad_sample;
end


function is_similar = histSimilar(histoModel,histTest,bins)
    is_similar = false;
    intersection = min(histTest,histoModel);
    res_intersection = sum(intersection(:))/sum(histoModel(:));
    res_dst_eucl = dist_eucl_histograms(histoModel, histTest, bins);
    res_dst_chi = dist_chisquare_histograms(histoModel,histTest);
```

```
    is_similar = is_similar | ((uint8(res_intersection > 0.51) + uint8(res_dst_eucl < 0.04) + u
end

function rgbNormalized = NormalizeRGB(I)
    I = double(I);
    max = I(:,:,1) + I(:,:,2) + I(:,:,3); %R+G+B
    max(max == 0) = 1 % preventing the problem of divide by 0 to happen
    rgbNormalized = I ./ max;
    % reparing once the problem has happend
    %rgbNormalized(rgbNormalized == Inf || rgbNormalized == NaN) = 0;
end

function dist_eucl = dist_eucl_histograms(h_A,h_B, n)
    dist_eucl = sqrt(sum(sum((h_A - h_B).^2)/n));
end

function dist_chisquare = dist_chisquare_histograms(h_A,h_B)
    % We only work with the ones that the sum is diferent than 0.
    diff_zero = (h_A + h_B) > 0;
    h_A = h_A(diff_zero);
    h_B = h_B(diff_zero);
    dist_chisquare = sum(sum( ((h_A - h_B).^2)./(h_A + h_B)));
end

function hist = getRBColorHistogram(I, bins)
    % I is an image in RGB
    aux = NormalizeRGB(I);
    I = [];
    I = aux(:,:,[1,3]);
    %range of I from  0 to 1
    [f, c, ~] = size(I);
    area = f*c;
    hist = zeros(bins, bins);
    for i=1:f
        for j=1:c
            %Classifying to the corresponding bin:
            pos_bin_red = min(floor( I(i,j,1) * bins) + 1,bins);
            pos_bin_blue = min(floor( I(i,j,2) * bins) + 1,bins);
            % Updating the bin of position bin_red and bin_blue
            hist(pos_bin_red,pos_bin_blue) = hist(pos_bin_red,pos_bin_blue) + 1;
        end
    end
    %Normalize the histogram
    if  area ~= 0
        hist = hist ./ area;
    end
end

%I es una imagen HS
function HSHist = getHSColorHistogram(I, bins)
    % I is an image in HSV
    H =  I(:,:,1);
    redHueMask =  (H >= 0.001 & H <= 10/180) | (H >= 160/180 & H <= 179/180);
    blueHueMask = H >= 0.5 & H <= 0.7;
```

```
    mask = I(:,:,2) > 0.1;
    H_selected = H(mask & (redHueMask | blueHueMask));
    S = I(:,:,2);
    S_selected = S(mask & (redHueMask | blueHueMask));
    I = [];
    I = [H_selected,S_selected];
    [f, c, ~] = size(I);
    area = f*c;
    HSHist = zeros(bins);
    for i=1:f
        %queremos dividir los valores de de hue y saturacion en 16 bins
        hBin = min( floor(I(i,1)*bins)+1, bins);
        sBin = min( floor(I(i,2)*bins)+1, bins);
        HSHist(hBin, sBin) = HSHist(hBin, sBin) + 1;
    end
    if area ~= 0
       HSHist = HSHist ./ area;
    end
end


function res = bbBarca(I)
    mida = size(I);
    Inorm = NormalizeRGB(I)*255;

    IblauBin = rgb2blauBin(Inorm);

    IvermellBin = rgb2vermellBin(Inorm);


    IBin = IblauBin | IvermellBin ;

    IBin = binTotal(IBin, IvermellBin, IblauBin);


    % indica el porcentaje del radio del disco respecto al lado mínimo de la
    % imagen
    porcentajeDisco = 0.5;
    radioDisco = round((min(mida(1), mida(2))*porcentajeDisco)/100);

    SE = strel('disk', radioDisco);
    IBin = imclose(IBin, SE);

    numPixels = numel(IBin);
    porcentajeDeAreaMinima = 1;
    numPixelsMinimo = round((numPixels*porcentajeDeAreaMinima)/100);
    IBin = bwareaopen(IBin, numPixelsMinimo);
    aux= regionprops(IBin, 'BoundingBox');
    if cellfun('length',struct2cell(aux)) > 0
          res = aux.BoundingBox;
    else
        res = [];
    end
end
```

```
function res = rgb2blauBin(I)
    threshold = 30;
    I = double(I);
    blau = [44, 71, 140];
    mida = size(I);
    res = false(mida(1), mida(2));
    for i = 1:mida(1)
        for j = 1:mida(2)
            val = [I(i, j, 1), I(i, j, 2), I(i, j, 3)];
            distBlau = heuDistColor(val, blau);
            res(i, j) = distBlau < threshold;
        end
    end
end

function res = rgb2vermellBin(I)
    threshold = 40;
    I = double(I);
    vermell = [145, 50, 60];
    mida = size(I);
    res = false(mida(1), mida(2));
    for i = 1:mida(1)
        for j = 1:mida(2)
            val = [I(i, j, 1), I(i, j, 2), I(i, j, 3)];
            distVermell = heuDistColor(val, vermell);
            res(i, j) = distVermell < threshold;
        end
    end
end

function res = binTotal(suma, binVermell, binBlau)
    mida = size(suma);
    res = false(mida(1), mida(2));
    for i = 1:mida(1)
        for j = 1:mida(2)
            if (suma(i, j))
                res(i, j) = sufficientWhite(i, j, binVermell, binBlau);
            end
        end
    end
end

function res = sufficientWhite(iOri, jOri, vermell, blau)
    mida = size(vermell);
    percentatgeThreshold = 1;
    % El numero de pixeles minimo que tiene que contar es proporcional al
    % numero de pixeles de la imagen
    threshold = (mida(1)*mida(2)*percentatgeThreshold)/100;
    percentatgeMidaMin = 20; % Determina el tamaño del area a tener en cuenta

    lengthWidth = uint32(mida(2)*(percentatgeMidaMin/100));
    lengthHeight = uint32(mida(1)*(percentatgeMidaMin/100));
    numBlaus = 0;
```

```
    numVermells = 0;
    for i = max(1, iOri-lengthHeight):min(mida(1), iOri+lengthHeight)
        for j = max(1, jOri-lengthWidth):min(mida(2), jOri+lengthWidth)
            if (vermell(i, j))
                numVermells = numVermells+1;
            elseif (blau(i, j))
                numBlaus = numBlaus+1;
            end
        end
    end
    res = numBlaus >= threshold && numVermells >= threshold;
end

function res = heuDistColor(c1, c2)
    res = sqrt((c1(1)-c2(1))^2 + (c1(2)-c2(2))^2 + (c1(3)-c2(3))^2);
end
```