VC Pràctica 1

Izquierdo Herrera, Jordi
Muntaner González, Joan Francesc

Març 2020

Índex

1 Introducció

L'objectiu d'aquesta pràctica és implementar en Matlab un algorisme d'identificació de samarretes de futbol. Donat un set d'imatges de jugadors de futbol, el programa ha de identificar quines imatges tenen una presència significativa de la samarreta del F.C. Barcelona. Per completar aquesta tasca aplicarem els coneixements obtinguts durant la primera part del semestre, implementant una comparació d'histogrames entre diversos models i les imatges a analitzar.

2 Algorisme Implementat

Primerament, hem elegit 3 imatges models de samarretes del F.C. Barcelona. Totes elles tenen les samarretes en primer pla, i ocupen la major part de la imatge. Per començar a analitzar una imatge, cridem la funció tryImage.m amb la imatge a analitzar i el model de color amb el qual volem analitzar la imatge, 'rgb' o 0 per el model RGB i 1 o 'hsv' per el model HSV.

2.1 tryImage.m

```
function p = tryImage(originalImage,colormodel)
    p = recursion(originalImage,-1);

    function p = recursion(im, prevSum)
        [p, votes, ~] = voteImage(im,colormodel);
        if (p == 1)
            return;
        end
        S = sum(votes, 'all');
        if (S>prevSum)
            [sx, sy , ~] = size(im);
            sx2 = floor(sx/2);
            sy2 = floor(sy/2);
            sx4 = floor(sx/4);
            sy4 = floor(sy/4);
            im1 = im(1:sx2,:,:);
            im2 = im(sx2:sx,:,:);
            im3 = im(:,1:sy2,:);
            im4 = im(:,sy2:sy,:);
            im5 = im(sx4:3*sx4,sy4:3*sy4,:);
            p = recursion(im1, S)|recursion(im2, S)|recursion(im3, S)
            |recursion(im4, S)|recursion(im5, S);
        else
            p = 0;
        end
    end
```

end

La funció `tryImage`, és una funció recursiva que va subdividint la imatge per intentar trobar exactament la regió amb les samarretes del Barça, i així fer una comparació més precisa. La entrada de Barca és la imatge, i la sortida és un valor booleà que indica si la imatge conté samarretes del F.C.Barcelona o no. Funciona de la següent manera:

1. Cridem la funció `voteImage` (explicada en el apartat 2.2) amb la imatge a analitzar. Si `voteImage` retorna p=1, s'ha conclòs que la imatge pertany al F.C.Barcelona i per tant retornem true i acabem la recursió. Aquest serà el nostre cas base. Si p=0, no s'ha conclòs i la recursivitat continua. També recollim de `voteImage` el nombre de vots (el resultat de cada model per cada mesura de distància) que ha obtingut la imatge.

2. Si el nombre de vots retornat per `voteImage` augmenta (anem passant la suma amb el paràmetre `prevSum`) suposarem que aquesta subimatge és més representativa que l'original i subdividirem la imatge en 5 subimatges. `im1` a `im4` són les meitats, superior, inferior, esquerre i dreta de la imatge, respectivament, i per tant, tenen la meitat de tamany. `im5` en canvi correspon a una imatge 4 vegades més petita (meitat d'amplada i d'altura) situada al centre. La següent figura ho il·lustra:

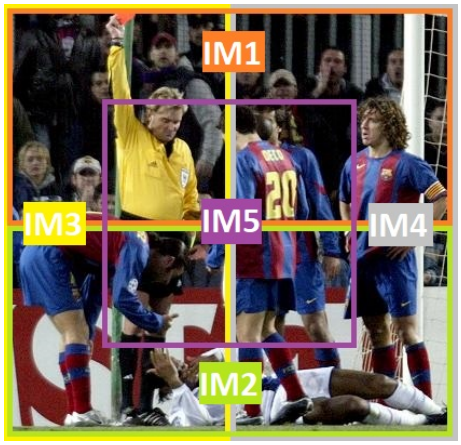

Figure 1: Divisió de les 5 subimatges

Cridem `recursion` per cada una de les subimatges. Si alguna d'elles retorna true, la funció retorna true. D'altre forma retorna false.

3. En canvi, si el nombre de vots retornat per `voteImage` és menor o igual al nombre de vots obtinguts en la recursió anterior, retorna false ja que aquesta subimatge no ha millorat els resultats de la imatge sencera i per tant de manera greedy suposem que no és una subimatge més representativa i per tant no ens interessa.

2.2 voteImage.m

```
% Casts votes of an image in relation to certain thresholds (obtained
% from percentils of the dataset to ease its finding).
%
% Parameters:
% colormodel: 'rgb',0 or 'hsv',1
% im: the image we want to evaluate
%
% Returns:
% p: final decision (0,1), it is one if at least for one of the distaces it
% surpasses the thresholds of the three models.
%
% votes: the vote for each model and distance (if the comparison with that
% model and distance surpasses the threshold or not).
%
% results: the raw values of the distances, before any comparison to
% thresholds has been made.
function [p,votes,results] = voteImage(im,colormodel)
    switch(colormodel)
        case{'rgb',0}
            % Histograms of each model
            m = matfile("models.mat");
            % the "raw" values given by this function we use to obtain an
            % adequate threshold from
            b = matfile("barcelona.mat").barcelona;
            % Parameters with the best performance according to our
            % experiments
            tChi = 48;
            tInter = 58;
        case{'hsv',1}
            % Same but with HSV
            m = matfile("modelsHSV.mat");
            b = matfile("barcelonaHSV.mat").barcelonaHSV;
            tChi = 60;
            tInter = 75;
    end
    H = m.H;
    h = imgaussfilt(histo(im,colormodel),1);
    barInter = b(:,1,:);
    barChi = b(:,2,:);
    BChi = prctile(barChi, tChi);
    BInter = prctile(barInter, tInter);
    N = size(H,1);
    results = zeros(2,N);
    votes = zeros(2,N);
```

```
    for k = 1:N
        hmodel = squeeze(H(k,:,:));
        %chisquare
        results(2,k) = histogramsCompare(hmodel,h,1);
        votes(2,k) = results(2,k)<= BChi(1,k);
        %intersec
        results(1,k) = histogramsCompare(hmodel,h,0);
        votes(1,k) = results(1,k)>= BInter(1,k);
    end
    M = max(votes(1,:),votes(2,:));
    p = sum(M,'all')>2;
end
```

La funció voteImage() és una funció que determina si una imatge conté samarretes del Barça, retorna un booleà que ho indica, els vots de cada model i distància i el resultats de cada model amb cada distància.

Primer de tot carreguem els models i thresholds adequats segons el model de color que fem servir. Els histogrames models estan continguts a `models.mat` (i `modelsHSV.mat` per HSV) (es poden obtenir usant la funció `histogramModels.m`. Els thresholds els carreguem a partir de els fitxers barcelona.mat: aquests contenen els valors (calculats prèviament amb `tryModels.m`) de comparar cada model amb cada imatge del barça del dataset (40 imatges) usant ambdues mesures de distància. Una vegada tenim aquests 40x2x3 valors, obtenim els thresholds particulars agafant els percentils d'aquests valors: `tChi` i `tInter`, obtinguts mitjançant experimentació.

Inicialitzades les dades de comparació, calculem l'histograma de la imatge a avaluar cridant la funció `histo` que ens retorna l'histograma 2D de la imatge amb la il·luminació normalitzada. A aquest histograma li apliquem un filtrat gaussià per tal de que afecti als bins veïns i facil·liti la comparació d'histogrames.

Finalment, comparem l'histograma obtingut amb els histogrames models. Per cada histograma model, es fa una comparació per Chi-Square distance i una comparació per intersecció d'histogrames usant la funció histogramsCompare.m. Com que tenim més d'un model i dues mesures de distància hem definit la següent funció de votació: per cada model i distància, es veurà si el resultat està dins del threshold definit. En cas positiu, quan estem sota del threshold a la chisqrt o per sobre del mateix en la intersecció, aquella distància i model "voten" 1. Així, en el nostre cas emetrem 6 vots, ja que tenim 2 mesures i 3 models (2*3); emetrem un resultat positiu (la imatge conté camisetes del Barça), si almenys una mesura de distància vota positiu a cadascun dels 3 models. És a dir, si el màxim dels vots de cada distància entre ells és 3. En cas contrari s'emetrà un resultat negatiu.

2.3 histo.m

```
% Calculates the histogram of image im accordidng to colormodel
function h = histo(im, colormodel)
    switch(colormodel)
        case{0,'rgb'}
            h = im2histo(im);
        case{1,'hsv'}
            h = im2histoHSV(im);
    end
end
```

La funció `histo` construeix un histograma 2D a partir de una imatge donada. Depenent de si es vol analitzar la imatge utilitzant RGB o HSV la funció cridarà `im2histo()` o `im2histoHSV()`.

2.3.1 im2histo.m (RGB)

```
% takes an image and returns the 2D-RB(normalised)-histogram
function h = im2histo(im)
    imN=im2norm(im);
    h=histcounts2(imN(:,:,1),imN(:,:,3),32);
end

% takes an image and returns the normalised image
function imN = im2norm(im)
    r=im(:,:,1);
    g=im(:,:,2);
    b=im(:,:,3);
    ilu=double(r)+double(g)+double(b);
    rN=double(r)./ilu;
    gN=double(g)./ilu;
    bN=double(b)./ilu;
    imN=cat(3,rN,gN,bN);
    imN=im2uint8(imN);
end
```

`im2histo()` construirà l'histograma d'una imatge amb un model de color RGB. Primerament, la imatge es normalitza utilitzant la funció `im2norm()`. Per normalitzar una imatge RGB, cal calcular la il·luminació de cada píxel(sumant les components R, G i B) i, seguidament, dividir aquest valor a les tes components. Seguidament, construirem un histograma 2D amb les components R i B utilitzant la funció de Matlab `histcounts2()` . Aquest histograma estarà dividit en un grid de 32x32 bins, dividint així els valors possibles de la combinació de R (vermell) i B (blau),

2.3.2 im2histoHSV.m (HSV)

```
% takes an image and returns the 2D-HS-histogram
function h = im2histoHSV(im)
    im=rgb2hsv(im);
    h=histcounts2(im(:,:,1),im(:,:,2),50); %eliminem la component value
end
```

`im2histoHSV()` fa el mateix que im2histo, però fent servir un model HSV. En primer lloc, s'utilitza la funció de Matlab rgb2hsv() per convertir la imatge en HSV. Per normalitzar la imatge, n'hi ha prou en eliminar la component V. Amb les components restants, es construeix un histograma 2D amb `histcounts2()` dividint els valors de la combinació de H i S en un grid de 50x50 bins.

2.4 histogramsCompare.m

```
% Compares histograms h1 and h2 with metric
function P = histogramsCompare(h1,h2,metric)
    switch(metric)
        case {0, 'intersect'}
            P = intersection(h1,h2);
        case {1, 'chisqrt'}
            P = chisqrt(h1,h2);
    end
end

function p = intersection(h1,h2)
    s1 = sum(h1,'all');
    s2 = sum(h2,'all');
    h1n = h1./s1;
    h2n = h2./s2;
    h = min(h1n,h2n);
    p = sum(h,'all');
end

function p = chisqrt(h1,h2)
    s1 = sum(h1,'all');
    s2 = sum(h2,'all');
    h1n = h1./s1;
    h2n = h2./s2;
    N = (h1n-h2n).^2;
    D = (h1n+h2n);
    P = N./D;
    p = sum(P,'all','omitnan')/2;
end
```

A la funció de comparació d'histogrames, com ja s'ha mencionat, hi hem introduït dues mesures de comparació. La distància chi-sqrt i la intersecció d'histogrames. Cal remarcar, que dos histogrames són més pròxims com més petita sigui la seva distància chi-sqrt i com més gran la intersecció (podem tractar la intersecció com un percentatge de semblança).

Cal remarcar també que abans d'aplicar les fórmules de cada mesura cal "normalitzar" els histogrames, per a que no siguin dependents del nombre de píxels de la imatge de la qual provenen. Per a fer això simplement dividim cada bin pel nombre valors totals de l'histograma.

2.4.1 Intersecció d'histogrames

Bàsicament consisteix en agafar per cada bin el mínim valor d'entre els dos histogrames. Intuïtivament representa com de gran o petita és la intersecció entre els dos histogrames. Si els dos histogrames són iguals, donarà el màxim valor, en canvi quan tenguin la intersecció buida, vol dir que no hi ha cap bin on hi hagi valors als dos histogrames i per tant hi assignem el mínim valor possible.

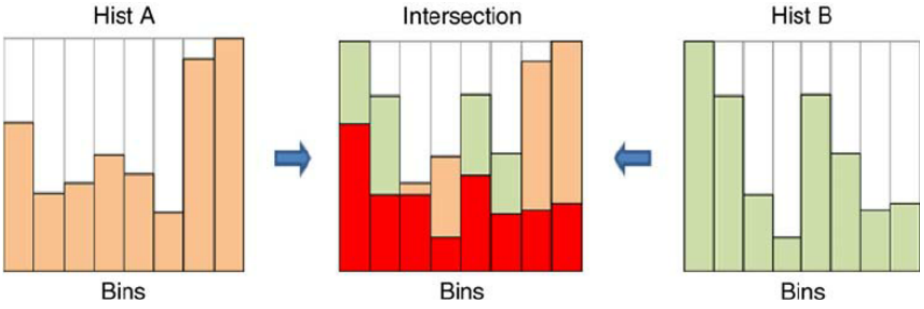

Figure 2: Exemple de funcionament de la intersecció d'histogrames

2.4.2 Distància Chi-Squared

Donats dos histogrames G i H la seva distància chisquared és defineix com:

$$d(G,H) = \frac{1}{2}\sum \frac{(G_i - H_i)^2}{G_i + H_i}$$

3 Experimentació i Resultats

Vist l'algorisme, està clar que té molts paràmetres que poden fer variar la correctesa de l'algorisme. Presentem un resum d'aquests paràmetres:

- Número de models i quins models particulars
  - Hem usat 3 models, obtinguts a partir d'agafar subimatges només de les camisetes del Barça de `soccer_data`.
- Número de bins dels histogrames
  - 32 bins per RGB i 50 bins per a HSV
- Tamany dels histogrames (quantes components)
  - Hem usat histogrames 2D, és a dir, hem descartat una de les components de color. A RGB al normalitzar la il·luminació, una component és depenent de les altres dues, i a HSV hem normalitzat la il·luminació eliminant la component v.
- Comparació amb la imatge sencera o amb subimatges
  - Hem fet proves amb la imatge sencera i després provant amb anar agafant subimatges més petites succesivament (detalls explicats a la funció `tryImage`).
- Mètriques de comparació entre els histogrames
  - Hem utilitzat dues mesures de comparació entre histogrames, la distància chi-sqrt i la intersecció d'histogrames.
- Funció de votació per decidir a partir de cada model i distància
  - Siguin Chi i Inter els vots de cada model respecte a una imatge, hem usat la funció $decision = max(Chi, Inter) > 2$;
- Thresholds acceptadors
  - Aquí és on més proves hem fet. Hem buscat els thresholds a partir dels percentils de les comparacions entre els models i les imatges del Barça de `soccer_data`. Així per a RGB, hem agafat els percentils 48 (chisqrt) i 58 (intersecció) i per HSV els 60 (chisqrt) i 75 (interseccio).

3.1 Millors resultats

Hem utilitzat el script definit per `tryData.m` per provar com de precís és l'algorisme:

```
% Tries the whole dataset with the current coded parameters
% Returns the following, in parethesis we indicate its range:
% totals: number of positives for each team (0-40)
% falsePos: percentage of false positives (0-100)
% falseNeg: percentage of false negatives (0-100)
% results: individual result for each image of the dataset (0,1)
function [totals, falsePos, falseNeg, results] = tryData(colormodel)
    path = './soccer_data/soccer/';
    equips = ['acmilan', 'barcelona', 'chelsea', 'juventus',
              'liverpool', 'madrid', 'psv'];
    results = zeros(7,40);
    for i = 1:7
        for j = 1:40
            num = string(j);
            if(j<10)
                num = strcat('0',num);
            end
            ipath = strcat(path,equips(i),'/',num,'.jpg');
            itmp = imread(ipath);
            results(i,j) = tryImage(itmp,colormodel);
        end
    end
    totals = sum(results,2);
    falsePos = (sum(totals,1) - totals(2,1))/(2.4);
    falseNeg = (40 - totals(2,1))/0.4;
end
```

La funció prova totes les imatges del DataSet (soccer_data) i retorna els resultats de la següent manera:

- totals: Retorna per cada equip, quants positius s'han retornat.
- falsePos: Retorna el percentatge de positius que haurien de ser negatius.
- falseNeg: Retorna el percentatge de negatius que haurien de ser positius.
- results: Retorna el resultat particular de cada imatge.

Si l'executem per RGB, els millors resultats (minimitzant la suma fp+fn) que hem obtingut han sigut un 10% de falsos positius i un 15% de falsos negatius. L'`error_rate` total pel conjunt del dataset, mitja de tots els errors, resulta de **10.71%**. Respecte a cada equip obtenim el següent (el resultat perfecte seria tot 0s i 40 al F.C.Barcelona):

A continuació mostrem els experiments realitzats per trobar els millors mètodes i els millors valors dels paràmetres :

Equip	nº Positius	`error_rate`(%)
AcMilan	12	30
F.C. Barcelona	34	15
Chelsea	4	10
Juventus	0	0
Liverpool	4	10
Madrid	0	0
Psv	4	10

Table 1: Millors resultats obtinguts amb `error_rate` total 10.71%

3.2 Evolució segons el Threshold

Un segon script que s'ha utilitzat per testejar, han sigut els `testFullDataSetRGB.m` i `testFullDataSetHSV.m`. N'analitzarem el de RGB, sense pèrdua de generalitat.

```
function R = testFullDatasetRGB
    %dataset de imatges
    global D;
    D = loadDataset;
    %results dataset sense recursio
    global barInter;
    global barChi;
    b = matfile("barcelona.mat").barcelona;
    barInter = b(:,1,:);
    barChi = b(:,2,:);
    %models
    global H
    m = matfile("models.mat");
    H = m.H;
    % tests with steps of one between 25 and 76 and stores in consecutive
    % values
    R = zeros(51,51,3);
    for i=1:51
        i
        for j=1:51
            j
            [~,fp,fn]=tryDatasetTRGB(i+24,j+24);
            R(i,j,1)=fp;
            R(i,j,2)=fn;
            R(i,j,3)=fn+fp;
        end
    end
    clear global;
end
```

`testFullDataSetRGB.m` itera per a totes les possibilitats dels thresholds (barInter i barChi), per trobar el valor més òptim. Després de fer córrer aquest script, hem obtingut els següents resultats, representats amb un gràfic de superfície, tant per RGB com per HSV:

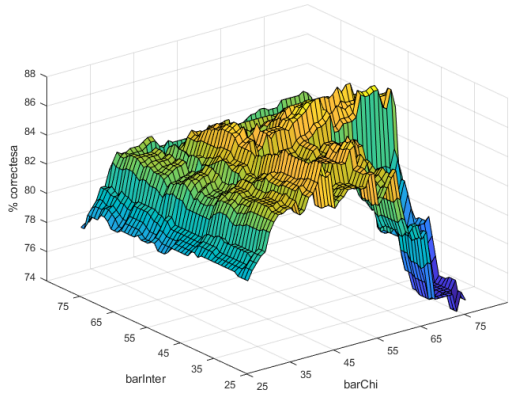

Figure 3: Paràmetres i correctesa amb RGB

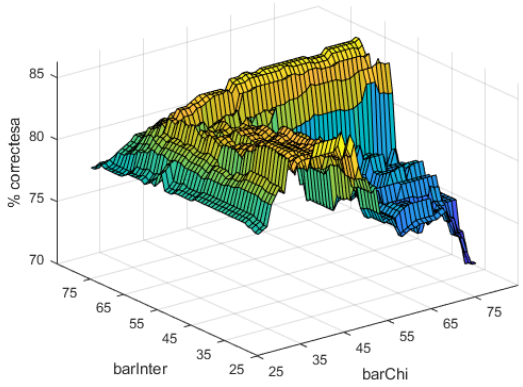

Figure 4: Paràmetres i correctesa amb HSV

Gràcies a aquestes proves, hem pogut esbrinar que els millors paràmetres(*) són barChi=48 i barInter=58 per RGB, i barChi=60, barInter=75 per HSV. (*)Cal dir que hem tractat de minimitzar la suma de falsos positius i falsos negatius.

3.3 Diferència entre Recursivitat i no Recursivitat

Volem observar com la recursivitat creada per `tryImage` millora la precisió dels resultats. Comparem els resultats de `tryImage` amb els de una iteració de `voteImage`, utilitzant el DataSet inicial:

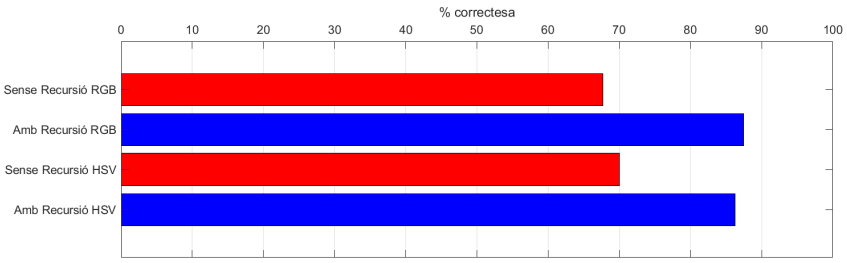

Figure 5: Comparació entre els diferents mètodes

Podem observar clarament com la recursivitat millora àmpliament el resultat, tant com per el model RGB com per el HSV. La recursivitat ajuda a detectar samarretes del F.C.Barcelona que no están en primer pla, són petites o estan menys definides:

Figure 6: Imatge positiva amb Recursivitat i negativa sense Recursivitat

La recursivitat també ajuda quan hi han deferents samarretes en una sola imatge i una d'elles es la del F.C.Barcelona. Sense recursivitat, l'algorisme crea un histograma de tota la imatge. La recursivitat pot crear una subimatge on els colors de la samarreta del F.C.Barcelona siguin els únics predominants en el histograma.

Figure 7: Imatge positiva amb Recursivitat i negativa sense Recursivitat

3.4 Exemples d'imatges amb mal resultats

En aquesta secció observem exemples d'imatges que l'algorisme ha detectat malament, ja siguin imatges que no contenen samarretes del F.C.Barcelona però l'algorisme detecta que sí, com imatges que sí que en contenen i l'algorisme no les detecta.
Del conjunt d'imatges donades inicialment, el AC Milan és el equip amb pitjors resultats, mentes que la Juventus i el Madrid són els equips amb millors resultats. Un dels problemes del conjunt del AC Milan és la semblança de colors en la samarreta:

Figure 8: Imatge amb fals positiu

Altres imatges són confoses per el fet de tenir dues samarretes diferents en una sola imatge, provocant histogrames similars a les del F.C.Barcelona:

Figure 9: Imatge amb fals positiu

Imatges del F.C.Barcelona també poden donar resultats erronis per diverses raons, com que estiguin mal enfocades, tenir un fons molt sorollós o no es vegin bé les samarretes:

Figure 10: Imatge amb fals negatiu

4 Funcions Utilitzades

4.1 Funcions pròpies

- Càlcul d'histograma i normalització
  - histo.m (amb paràmetre colormodel)
  - im2histo.m (RGB)
  - im2norm.m (RGB)
  - im2histoHSV.m
- Decidir sobre una imatge (amb subdivisions).
  - tryImage.m (amb paràmetre colormodel)
  - barcaRGB.m
  - barcaHSV.m
- Decidir sobre una imatge (sense subdivisions).
  - voteImage.m (amb paràmetre colormodel)
  - testImageRGB.m
  - testImageHSV.m
- Comparació d'histogrames
  - histogramsCompare.m
- Inicilització dels histogrames dels models
  - histogramModels.m (amb paràmetre colormodel)
  - initModels.m (RGB)
  - initModelsHSV.m
- Precàrrega de les imatges
  - loadDataset.m
- Càlcul valors de `soccer_data` (per al threshold)
  - tryModels.m
- Prova les imatges de `soccer_data` amb els paràmetres assignats
  - tryData.m (amb paràmetre colormodel)
  - tryDatasetRGB.m
  - tryDatasetHSV.m
- Testing dels paràmetres tChi i tInter

  - testFullDataset.m (amb paràmetre colormodel)
  - testFullDatasetRGB.m
  - testFullDatasetHSV.m
  - Funcions per provar sobre `soccer_data` sense paràmetres assignats
    * tryDatasetTRGB.m
    * tryDatasetTHSV.m

4.2 Funcions de Matlab

- histcounts2
- imgaussfilt
- rgb2hsv
- sum
- max
- prctile
- floor
- double
- matfile
- im2uint8
- zeros
- imread
- strcat
- cat
- shiftdim
- string
- resize
- find