1 Matlab correction

1.1 Geometrical functionals

The Crofton perimeter and Feret diamaters have been already computed in the tutorial about integral geometry. For computing the perimeter, we use the regionprops function.

```
% classical perimeter
if (~islogical(I))
    perim = regionprops(I>100, 'Perimeter');
else
    perim = regionprops(I, 'Perimeter');
end
p = perim.Perimeter;
```

The Crofton is evaluated with the use of 4 directions.

```
I=double(I);
inter = zeros(4,1);
for i=1:4
    I2=imrotate(I, 45*(i-1));
    h=[1 -1];
    I3=conv2(I2, h, 'same')>0;
    inter(i) = sum(I3(:));
end

% formula of the Crofton perimeter
crofton = pi/4*(inter(1)+inter(3) + (inter(2)+inter(4))/sqrt(2));
```

The Feret diameter corresponds to the projected length in a given direction. The following code uses a angular step of 30 degrees. It computes the maximum and the minimum at the same time.

```
function [d,D] = feret(I)

d=max(size(I));
D=0;
for a=0:30:179
    I2 = imrotate(I, a);
    F=max(I2);
    mesure=sum(F>0);

    if (mesure<d)
        d=mesure;
    end
    if (mesure>D)
```

```
        D=mesure;
    end
end
```

The radius of the inscribed circle can be computed as:

```
function r=InscribedCircleRadius(I)
dm = bwdist(~(I>0));
r=max(dm(:));
```

1.2 Morphometrical functionals

The different morphometrical functionals can be easily computed as ratios of geometrical functionals.

1.3 Shape diagrams

The following process gives 3 shape diagrams for the whole Kimia image database:

```
name={'apple-' 'bone-' 'camel-'};
couleur={'r+' 'g*' 'bo'};
format='.bmp';
indices=1:20;

% number of diagrams:
N_diag=3;
x=zeros(N_diag, length(name)*length(indices));
y=zeros(N_diag, length(name)*length(indices));

xlabs={'\omega/d' '\omega/d' '2*r/d'};
ylabs={'2*r/d' '4A/(pi*d^2)' 'P/(pi*d)'};

% computation of the functionals for all the images
for n=1:length(name)
    for i=indices

        I = imread([name{n} num2str(i) format]);
        [omega d] = feret(I);
        [crofton, P] = perimetres(I);
        r = rayonInscrit(I);
        stats = regionprops(I>0, 'Area');
        ii = (n-1)*length(indices)+i;
        x(1,ii)=omega/d;
        x(2,ii)=omega/d;
```

```
        x(3,ii)=2*r/d;

        y(1,ii)=2*r/d;
        y(2,ii)=4*stats.Area/(pi*d^2);
        y(3,ii)=P/(pi*d); % values > 1, non-convex objects

        % progress bar
        waitbar(((n-1)*length(indices)+i)/(length(name)*length(indices)))
            ↪ ;
    end
end
```

The visualization is done via the following code and displayed in Fig.1.

```
% visualization of the 3 diagrams with a specific color for each category
for j=1:N_diag
    figure(); hold on
    for i=1:length(name)
        i1=length(indices)*(i-1)+1;
        i2=length(indices)*i;
        plot(x(j,i1:i2), y(j,i1:i2),[ couleur{i} ]);

    end
    legend(name);
    xlabel(xlabs{j});
    ylabel(ylabs{j});
end
```

1.4 Shape classification

The MATLAB® function kmeans is used for classifying the shapes into 3 classes:

```
X1=x(1,:);
X2=x(2,:);
X3=x(3,:);
Y1=y(1,:);
Y2=y(2,:);
Y3=y(3,:);
idx1 = kmeans([X1',Y1'],3);
idx2 = kmeans([X2',Y2'],3);
idx3 = kmeans([X3',Y3'],3);
```

The classification accuracy for each shape diagram is then computed as a percentage of correctly positioned shapes.

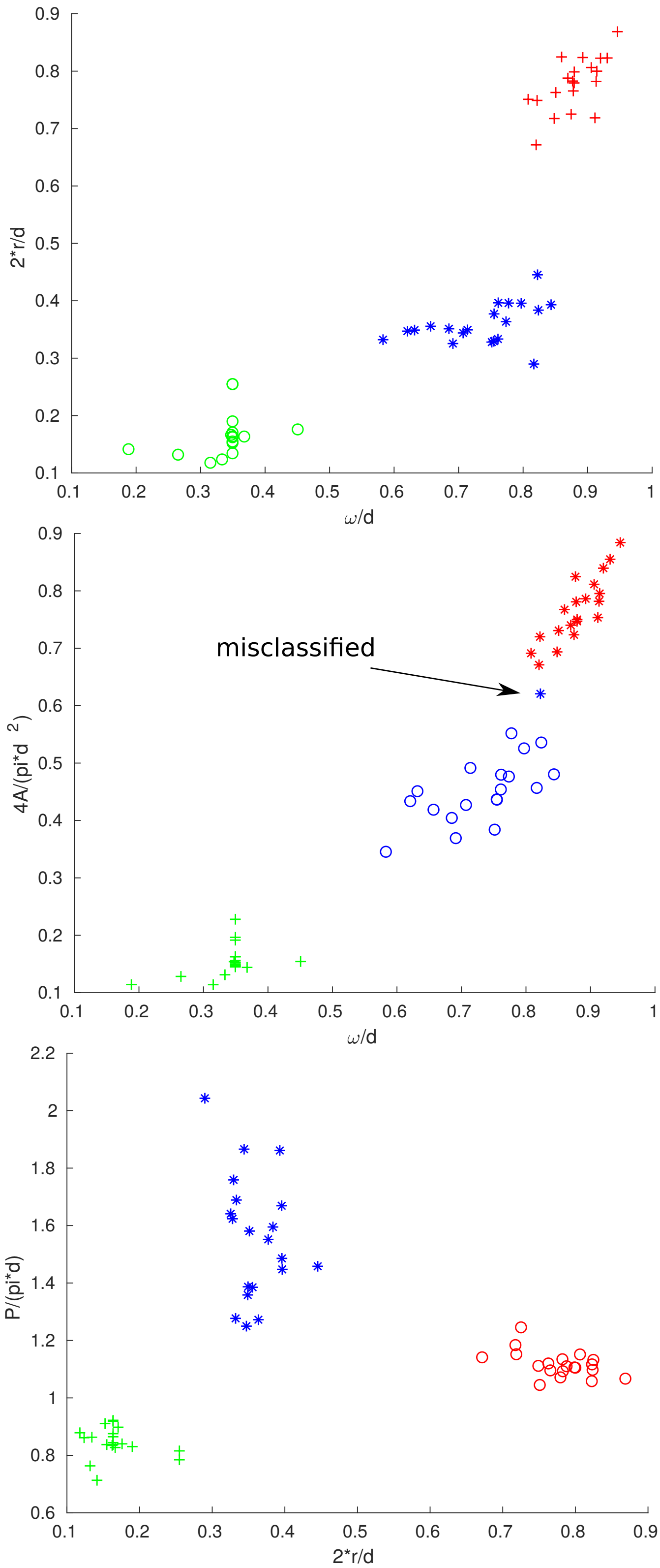

Figure 1: Three shape diagrams for the three binary images from the Kimia database, and their classification by the kmeans algorithm. The color denotes the real class, and the shape denotes the results of the kmeans algorithm.

```
accuracy1 = (sum(idx1(1:20)==mode(idx1(1:20))) + sum(idx1(21:40)==mode(idx1(21:40))) + sum(idx1(41:60)==mode(idx1(41:60))))/60*100
accuracy2 = (sum(idx2(1:20)==mode(idx2(1:20))) + sum(idx2(21:40)==mode(idx2(21:40))) + sum(idx2(41:60)==mode(idx2(41:60))))/60*100
accuracy3 = (sum(idx3(1:20)==mode(idx3(1:20))) + sum(idx3(21:40)==mode(idx3(21:40))) + sum(idx3(41:60)==mode(idx3(41:60))))/60*100
```

with the following results. This quantifies the error done in second diagram (Fig.1).

Command window

```
accuracy1 = 100
accuracy2 = 98.3333
accuracy3 = 100
```