Internship Report
Image segmentation on tooth radiography

Louis VIOT, Quentin BERNARD

September 29, 2014

Contents

1 Abstract and acknowledgments

Abstract

Dental image processing, like teeth extraction from dental radiography, could have many applications in the forensic field for dental recognition or in dental diagnostic. The main problem is that there is yet not any automatic tool extracting tooth from a dental radiography, so our goal here was to find a method to extraxt those teeth.
Since radiography are black and white picture, a pixel is represented in the following by its coordinates in the picture and by its gray intensity. Segmentation method can be classify in two categories : in the first category, methods are using region's statistics to cluster parts of the radiography, in the second category, methods are using edge detection.
Our first research were about the active contour method (method of the first categorie) since traditional edge detection method weren't effective at all on picture like radiography. We were to found a stopping criterion and had to improve the method to make it work on radiography. What we found out is that, since radiography are often really different, the active contour method could work perfectly on one radiography with certain parameters but could not work at all on another radiography with the exact same parameters. That's why, in section 3, we tried to create pre-treatment method aiming at making the active contour method works on every radiography with the same parameters, to make an automatic method.
Finally, we tried to gather all of our method to create an automatic tool in section 4.

Acknowloedgments

As our first research internship, we would like to thank M.Balsa for his guidance and kindness, not only during the work, but also outside of the university. Thanks to his advices and his company, we really enjoyed our time in Bragance.

2 Chan-Vese algorithm

The Chan-Vese algorithm is an active contour method. In active contour methods, the goal is to minimise an energy along a curve, by evolving that curve in a descent direction for the energy. Energy is computed on image data and curve's regularity. Many models exist for the curve energy. The Chan-Vese model has the advantage to not be based on edge detection, so can detect objects without distinct edges.

2.1 The Chan-Vese model for the energy

The energy to minimise in Chan-Vese method is :

$$\begin{aligned} F(c_1, c_2, C) = \alpha.\text{Length}(C) + \beta.\text{Area}(inside(C)) \\ + \lambda_1 . \int_{inside(C)} |u_0(x,y) - c_1| dxdy \\ - \lambda_2 . \int_{outside(C)} |u_0(x,y) - c_2| dxdy \end{aligned} \tag{1}$$

Where C is a closed curve (which should fit the target object), u_0 is the gray level function from the picture, c_1, c_2 respectively the inner and outter gray level means.

The term $\alpha.\text{Length}(C)$ ensure smooth of the curve, as we minimise its length. $\beta.\text{Area}(inside(C))$ is also about regularity.

The term $\lambda_1 . \int_{inside(C)} |u_0(x,y) - c_1| dxdy - \lambda_2 . \int_{outside(C)} |u_0(x,y) - c_2| dxdy$ should be minimum when the curve fits at best the object. An intuitive way to see it : for an monochromatic object on an monochromatic background, this term will be null if the curve fit the object.

2.2 Minimise the energy

This part is mathematical. It aims to exhibit a way to minimise F algorithmically.

We need a way to calculate C. So we redefine C implicitly by $\phi : \mathbb{R}^2 \to \mathbb{R}$ such that :

$$\phi = \begin{cases} 0 & (x,y) \in C \\ < 0 & (x,y) \in outside(C) \\ > 0 & (x,y) \in inside(C) \end{cases}$$

Then we can write the energy F :

$$F(c_1, c_2, C) = \alpha. \int_\Omega \delta_0(\phi(x,y))| \bigtriangledown \phi(x,y)|dxdy + \beta. \int_\Omega H(\phi(x,y))dxdy$$
$$+ \lambda_1. \int_\Omega |u_0(x,y) - c_1|H(\phi(x,y))dxdy \quad (2)$$
$$- \lambda_2. \int_\Omega |u_0(x,y) - c_2|(1 - H(\phi(x,y)))dxdy$$

Where H is the Heaviside function and δ_0 the Dirac distribution.

Both function are regularised, respectively by H_ϵ and δ_ϵ (in the algorithm we need only to compute δ_ϵ which is done by taking a narrow band around ϕ)

This formulation allows us to apply Euler-Lagrange equation, which give F derivative relatively to ϕ

So we have a descent direction for ϕ. Parameterising ϕ descent by t, we get :

$$\frac{\partial \phi}{\partial t} = \delta_\epsilon(\phi) \left[\mu.div \left(\frac{\bigtriangledown \phi(x,y)}{| \bigtriangledown \phi(x,y)|} \right) - \nu + \lambda_1(u_0 - c_1)^2 - \lambda_2(u_0 - c_2)^2 \right] \quad (3)$$

For more details see [1].

2.3 Algorithm, stopping criterion

The algorithm is then the following :

- Initialize ϕ
- Compute c_1 and c_2
- Iterate ϕ in descent direction
- If convergenced then stop else go back to second step

As the algorithm is actually a conjugate gradient one, we could check the convergence by checking stationnarity of ϕ. Though it appears that at convergence ϕ slighty oscillate.

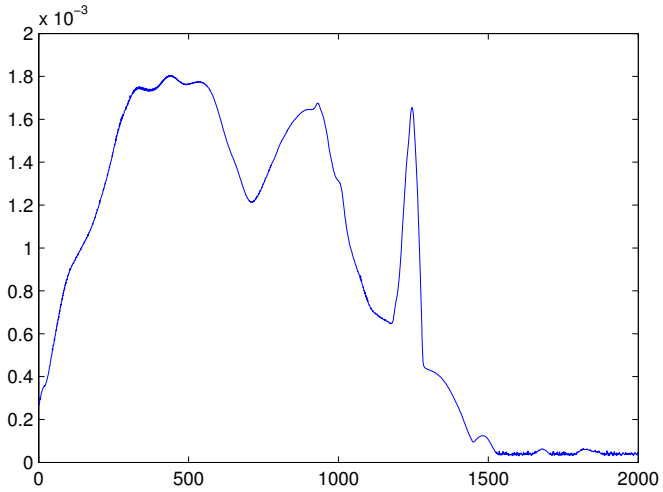

Figure 1: $||\phi_{n+1} - \phi_n||$ according to the number of iteration, for rx1 picture

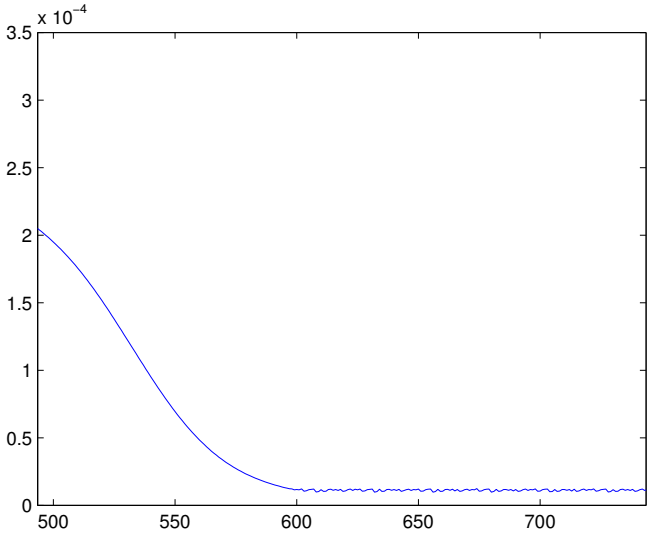

Figure 2: $||\phi_{n+1} - \phi_n||$ according to the number of iteration, for rx1 picture

$S_1 = ||\phi_{n+1} - \phi_n||$ can't be taken as a stopping criterion. This value stabilise at around 10^-4 for rx1 (see figure 1), 10^-5 for rx2 (see figure 2). So it is hard to define a tolerance. Then we decided to choose as a stopping criterion the derivative of S_1, filter by a low pass filter (to suppress oscillation effect, see figure 3). The algorithm stop if $\frac{dS_1}{dn} < tol$ over 20 iterations.

Oscillations can be explain by the fact that c_1 and c_2 depends on ϕ, but doesn't appear in descent term, so a sightl error is introduced which prevent the algorithm to totally stabilise.

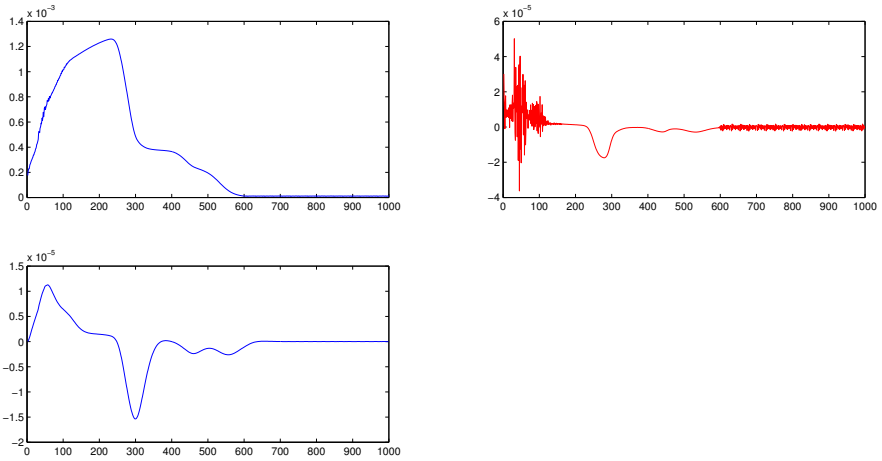

Figure 3: From left to right : S_1, its derivative, and its filtered derivative

2.4 Tests

As Euler-Lagrange equation can only ensure a local minimum, the algorithm depends on initialisation. It also depends on coefficients α relative to $_1$ and λ_2. We set $\lambda_2 = 1$ for the tests. Figures in annexes shows the result depends on whole parameters. Moreover goods results may be obtain for different parameters on two pictures. The work to be described further is try to automatise the algorithm (reduce or even suppress the parameters dependency).

2.5 Local segmentation

See section 4.1 for the `MatLab` *code.*
A way to really improve the active contour method for more heterogeneous picture, like radiography, is to use local statistics instead of global statistics in the algorithm. Indeed, we can see in figure 4 that the bottom of the tooth has almost the same shade of gray than some part of the gum. So the global method won't be able to distinguish those two parts of the radiography. The key here is to not examine those two parts together, and instead, examine each pixel of the radiography locally. In other words, at each iteration of the algorithm, instead of having statistics about the whole picture, we create a

small ball centred on the current pixel and calculate the same statistics but inside the ball. For further information, see [1].

Figure 4: Interest of the local active contour method : bottom of the tooth has the same shade of gray as some part of the gum

In the following (see the next two figures), we create another initialisation for the active contour method by using two ellipses, one around the tooth and one between the two roots. So that if we take the area between the two ellipses, we can have a pretty good initialisation. Here is an example of the improvement brought by the local method compared to the global method (we used the exact same parameter and the same number of iteration):

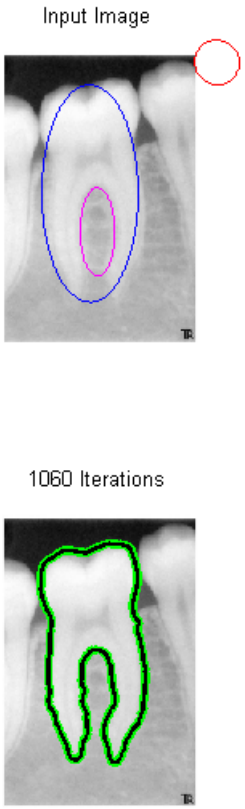

Figure 5: Segmentation with the local method : Red circle has the same radius as the ball used in the algorithm. The two ellipses are used for the initialisation.

As we can see, the segmentation is here almost perfect with the local algorithm. Segmentation with the region based active contour method is behaving as expected : since the tooth shade of gray is almost the same as the gum, the algorithm cannot separate them (see figure 6). However, local algorithm is really slow since we have to consider each pixel around the zero level set of ϕ and calculate statistics of pixels in the ball centred on the first pixel. The total execution time of the local algorithm was about 20 minutes on this high definition radiography while the global method execution time was barely 30 seconds. To improve the local algorithm, we have to use another way to implement the level set function ϕ. The best is the sparse field method proposed by Whitaker (see [2] for more details). The author of [1] implemented this method on the local algorithm and it did save a lot of computations (see [3]).

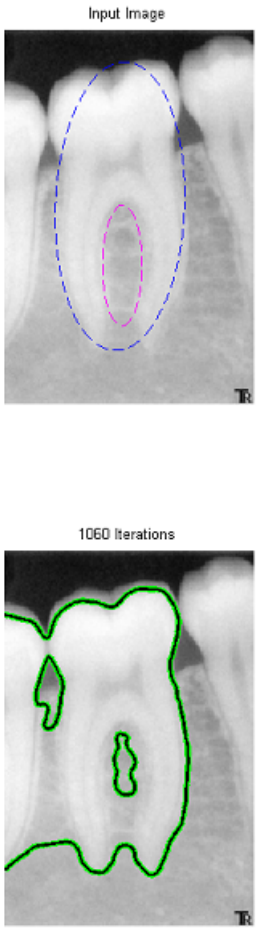

Figure 6: Segmentation with the global method : bad segmentation between the two teeth.

3 Differents way to decrease parameters dependency

We experimented some ways to reduce initialisation, α and λ_1 dependency. Those are described in the following sections.

3.1 Clustering initialisation

See section 4.2 for the `MatLab` *code.*
Clustering aims at sorting pixel of the radiography into a certain number of cluster. If we look closely to the radiography histogram (see figure 7), we can see three peaks : the tooth (white), the gum (gray) and the background (black).

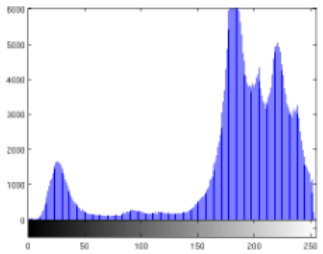

Figure 7: rx2 radiography histogram

As we have seen previously, the initialisation in the active contour method is very important and it could be very interesting to have a mask already almost on the tooth. So clustering the radiography in three parts in order to extract the most we can of the tooth could make the Chan-Vese method works better. In the following, we will talk about the clustering algorithm we have been working on : the k-means and the Otsu algorithm.

Quick explanation of the k-means algorithm

We have a set of $(x_i)_{i=1..n} \in \mathbb{R}^p$, we want to classify them in k clusters. We note $\Pi = (\pi_j)_{j=1..k}$ the k clusters and $m_j = \frac{\sum\limits_{l \in \pi_j} x_l}{|\pi_j|}$ the mean or centroid of the cluster j . The aim of k means is to minimise the sum of the distance between each point in the cluster and its centroid on every cluster :

$$\inf_{\Pi} \sum_{j=1}^{k} \sum_{l \in \pi_j} \|x_l - m_l\| \tag{4}$$

The k means algorithm is a heuristic and is described in the following :

- Start with k centroid and calculate the corresponding error.
- For each pixel in the radiography, find the closest centroid and put that pixel in the centroid's cluster.
- Calculate the new error : if the difference between the new error and the old is less than tol, then stop, else go back to the second step

As we can see, the k means is really dependent on the initialisation : different runs of k means can have different outputs. The Global K-means algorithm is another algorithm using the local k mean descirbed above and does not depend on the initialisation. The basic idea of global K means is that the best solution for clustering an image into M clusters can be found from the solution of the M-1 clustering problem. However, Global K means cost a lot more in term of space and time. For more information about Global K means, see [4]. The multi-level thresholding Otsu method is based on the Otsu method which minimise a certain variance to sort pixel in different cluster. It is equivalent to K means algorithm, that's why we won't use it in the following.

K means with radiography

By using K means, or global K means, we can now cluster pixel of the radiography in three parts and extract the part corresponding to the tooth. However, we have to choose a norm for formula 4. We have choose to compare three different norms (we note (i,j) coordinates of a pixel and I_{ij} its gray intensity) :

- The L_1 norm or City Block norm : $|I_{ij} - I_{i'j'}|$
- The L_2 norm or Euclidian norm : $\|I_{ij} - I_{i'j'}\|$
- Since the last two norms only consider the gray intensity, we could think of a norm taking the geometry into account.

$$\alpha \times d((i,j),(i',j')) + \beta \times \frac{|I_{ij} - I_{i'j'}|}{255} \tag{5}$$

where α and β are parameters

Several tests have shown that the third norm isn't really interesting since the best value for α is 0.05 and 0.95 for β, so geometry isn't important in this problem. Moreover, the first norm is often better for what we have to do. For instance, on the figure 8 below, a biggest part of the tooth is detected (left) than with the second norm (right).

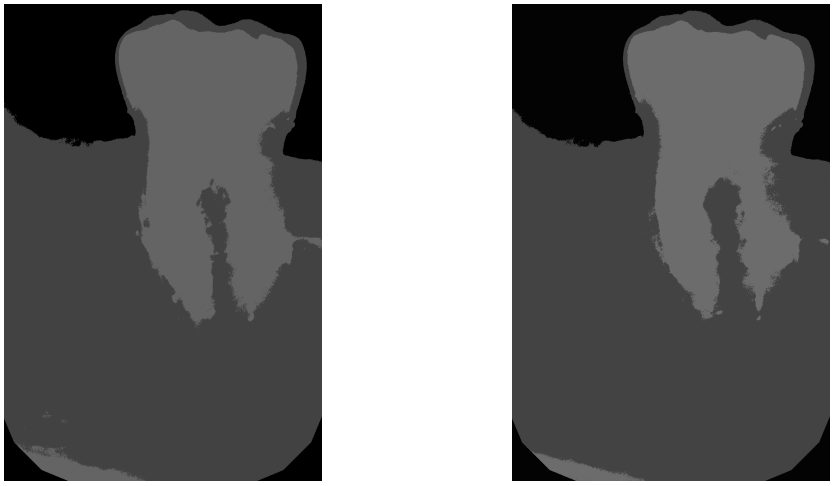

Figure 8: Run of kmeans with norm 1 (left) and norm 2 (right)

In the following, we only use the first norm (city block norm or L_1 norm). We choose to use always four clusters to extract the tooth instead of three : indeed, in some radiography, there is a part inside the tooth with a different shade of gray. So if we take only three clusters, we might miss the tooth (see figure 9)

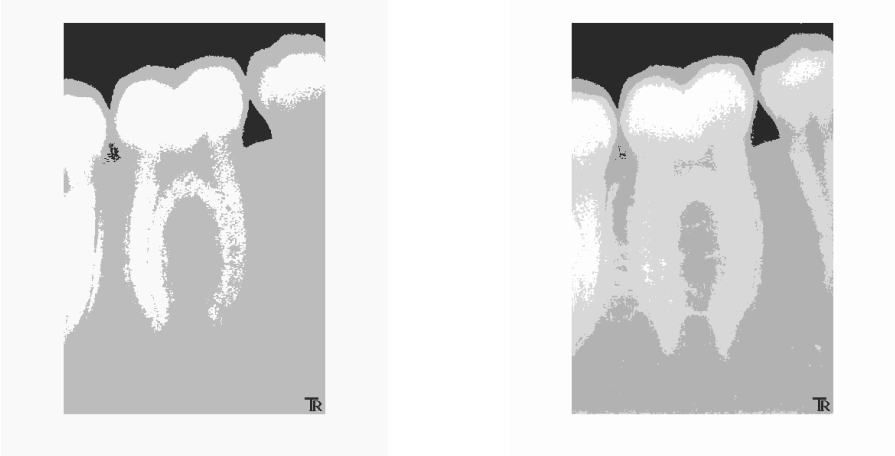

Figure 9: Number of cluster : 3 in the left and 4 in the right

For the active contour method initialisation we need a black and white mask. After clustering the radiography in four parts, we know that the two lightest part are the part corresponding to the tooth since the two others are the gum and the background : those two parts are going to be white, the others will be black. Here is the output of the execution of the algorithm on different radiography :

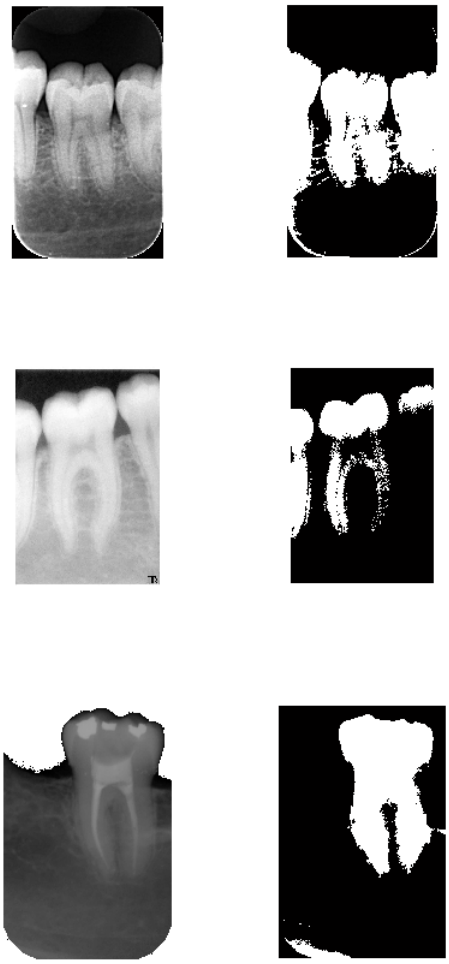

Figure 10: Extraction of the tooth : in the left the original radiography, in the right the mask.

As we can see, clustering is not perfect : there are still a lot of noises where the gum is. That's what motivated us to use the PCA method to extract the gum before using clustering in section 3.3.1.

3.2 Curve guidance

According to figures the α parameter influence the final shape. A too big α will make the algorithm extract a part of a tooth only, when a too small α lead to an unsmooth curve (which might includes gum's nerves), or will extract many tooth.

In order to fix this issue, we introduce a new term in Chan-Vese energy

F. The idea is to add a gravity energy term, guiding the curve into a hole (some holes) around the tooth (by minimising the gravity energy).

Guide the curve to a predefine contour

We assume we have a contour C_{ap} approximately fitting the curve. We then add a surface gravity energy to Chan-Vese energy, compute over $Inside(C_{ap})$ We had to add to (1) the term :

$$\begin{aligned} E_p &= \frac{1}{\text{Area}(C)} \int_{inside(C)} e(x,y)dxdy \\ &= \frac{1}{\text{Area}(C)} \int_{\Omega} e(x,y)H(\phi(x,y))dxdxy \end{aligned} \tag{6}$$

Where $e(x,y)$ is a surface energy field, negative inside C_ap (push the active contour to the outside), positive outside (push the active contour to the inside).

We compute the Area(C) at each iteration (as for c_1 and c_2).
Then applying the Euler-Lagrange formula to 6 add to 3 the simple term $e(x,y)$ We compute $e(x,y)$ as the square of the distance to (x,y) from C_{ap}

Figure 12 shows the Chan-Vese algorithm application without hole, figure 11 the computation with a rectangular hole around the tooth. The result isn't better (as the hole is rectangular), but the goal (preventing the curve to leave the tooth for a too small alpha) is reached. Moreover the computation speed increase.

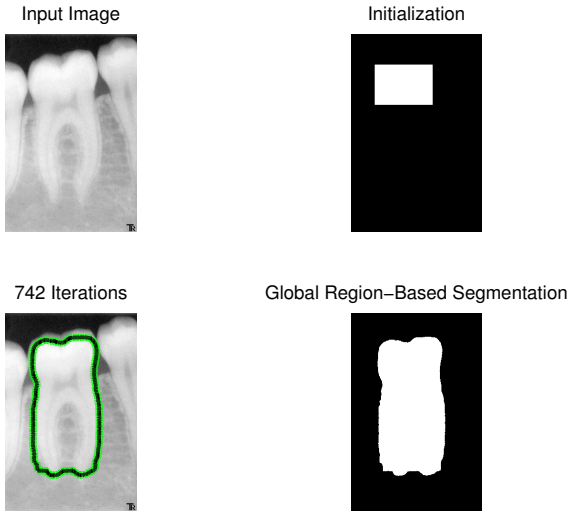

Figure 11: Computation with a rectangular C_{ap} hole around the tooth

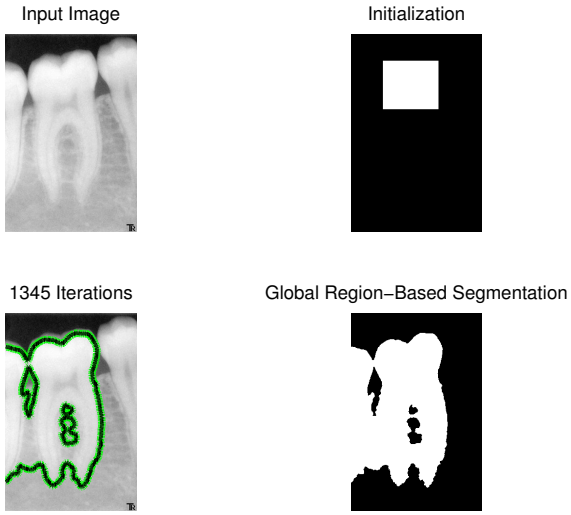

Figure 12: Computation without the rectangular C_{ap} hole around the tooth

Guide the curve to control points

The idea is the same than in the previous part, but here we compute a linear gravity energy along C. We had to (1) the term :

$$
\begin{aligned}
E_p &= \frac{1}{\text{Length}(C)} \int_C e_l(x,y) dC \\
&\approx \frac{1}{\text{Length}(C)} \int_\Omega e_l(x,y) \delta_\epsilon(\phi(x,y)) | \bigtriangledown \phi(x,y) | dxdxy
\end{aligned}
\tag{7}
$$

Where $e_l(x,y)$ is a linear energy field, such as $M = min(e_l(x,y))$ is the set of required control points. Then the curve should be "attracted" by those control points.
The energy (7) add to (3) the term :

$$
\frac{\partial e_l}{\partial x} . \frac{\partial \phi}{\partial x} + \frac{\partial e_l}{\partial y} . \frac{\partial \phi}{\partial y} \tag{8}
$$

The implementation doesn't work so far, so we have no tests to propose. This method is only reported here for information.

3.3 Image pre-treatment

Texture analysis by PCA

See section 4.3 for the `MatLab` *code.*
As seen previously, the gum is causing a lot of trouble in active contour algorithm. In radiography of our database, the gum between teeth is like noise preventing the good behaviour of almost every of our algorithm (as shown below in figure 13 showing gum noises between teeth). Thus, our goal here is to get rid of the gum by using some of its characteristics.

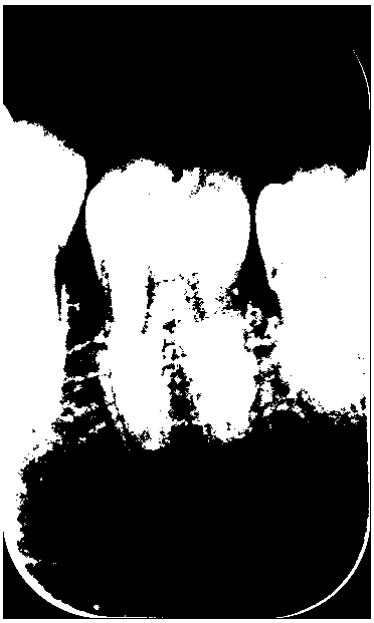

Figure 13: Noises between teeth caused by gum in k-means clustering

Our first idea was to use the gum pattern to recognise and search the gum in radiography. To do that, we are using principal component analysis. The PCA is based on a pretty simple idea : starting with a set of p points in $\mathbb{R}^n$ (or p subjects defined by n varibles), we want to find vectors explaining the maximum variability of those points. Those vectors are actually really simple to find. Indeed, if we note X_c the p x n matrix filled with the previous points minus their means, then it is easy to show that those vectors are eigenvectors of the matrix $X_c^t X_c$, each vector explaining a certain variability depending on the size of the associate eigenvalue.
We will be using the PCA on the gum to find its characteristics and to be able to recognise it. To do so, we are applying the same algorithm on each picture of our database :

- Select a square in the gum pattern
- Split this square in p little bloc of size $\omega \times \omega$. ω must be big enough to have enough pixel and small enough to not have big basis.
- Each bloc of ω^2 variables is a subject in the PCA so that the matrix sized used is $p \times \omega^2$. Then, the PCA gives us a basis of bloc of ω^2 vectors of size ω^2 explaining the maximum variability of those blocs.

We can then use this basis to rebuild the whole radiography by splitting the image in blocs of size ω and by projecting all of those blocs on the first vector of this basis, since it's the one explaining the most of the variability.

We know how to build a basis of bloc for each pattern of the radiography, hence, we do that for the gum, the tooth and the background. In our problem, we have to recognise and sort all the bloc by their texture. Thus, we select a bloc of the picture. In the following, we note I_{proj} the projection of the bloc on the first k basis vectors, $(B_i)_{(i=1..nText)}$ the nText pattern bloc basis where nText is the total number of texture. Thus, the projection of the bloc I on the basis B_j is :

$$\exists(\alpha_1, \alpha_2, ..., \alpha_k) \in \mathbb{R}^k, I_{proj} = \sum_{j=1}^{k} \alpha_j \times B_j(:, j) \tag{9}$$

To sort the bloc I in the right texture, we had two ideas :

- We project the bloc on each pattern bloc basis B_j, we calculate the L_2 norm error $err(j) = \|I_{proj} - I\|$ and we choose the one basis minimising this error : $min_{j=1..nText} err(j)$.
- We project the bloc on each pattern bloc basis B_j. Since the projection I_{proj} must not be good on all of the basis except the right one, the projection coefficient must be small on the right texture basis and big on wrong texture basis. Then we choose the basis j such as : $min_{j=1..nText} \sum_{i=1}^{k} |\alpha_i|$

After several tests, we found out that the second criterion is the best to sort bloc by their texture. For instance in figure 14, we have three textures and we use the algorithm described above to sort each bloc of the picture by their texture. In the third and fourth part of the MatLab figure, each bloc of the original picture has a gray shade : it is white if it belongs to the texture in the right of the picture, gray if it belongs to the middle texture, dark gray if it belongs to the left texture, depending on the criterion used to sort. In the third part, we use minimisation of the projection coefficient criterion, in the fourth part, we use the minimisation of the error criterion. Since we know the obvious answer of the sorting, we can compare which method is the best.

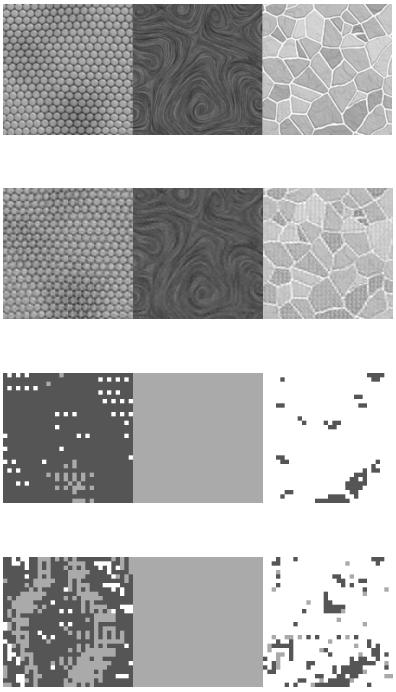

Figure 14: Sorting bloc by their texture with PCA

Once we have the criterion to sort bloc by their texture, we can build a database of gum texture to be able to sort the gum of every kind of radiography. We started by building it with five radiography, and we tried to run the sorting algorithm on other radiography found on the internet. Here are the results of our tests (in white the gum, in black the other part) :

Figure 15: in white the gum, in black the other part

The sorting is not really well working when we use data from our database. This might be caused by too many different shade of gray in the radiography used to create the database since the sorting is pretty well working (see figure 16) when using basis extracted directly from the radiography, and not from other database basis.

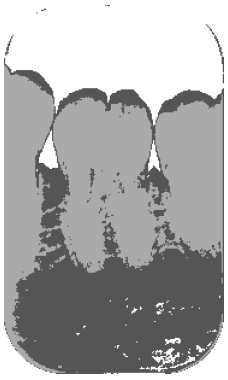

Figure 16: Sorting using only basis calculated from the radiography

Different modification on the original image

We tried different simple pretreament on the image. Only two showed off some results. The first is to black the image away from an aproximate mask fitting the tooth. As Chan-Vese need to compute inner and outter intensity means, the goal is to avoid some perturbation due to picture's parts which have the same color as the tooth. Figure (17) show the black rx2 image. Figure (19) and (18) show how this can reduce the α dependency. Indeed as

the curve isn't attracted by the others tooth or nerves in gum, the curve is smoothest, without the need to increase α

Figure 17: Black rx2.jpg

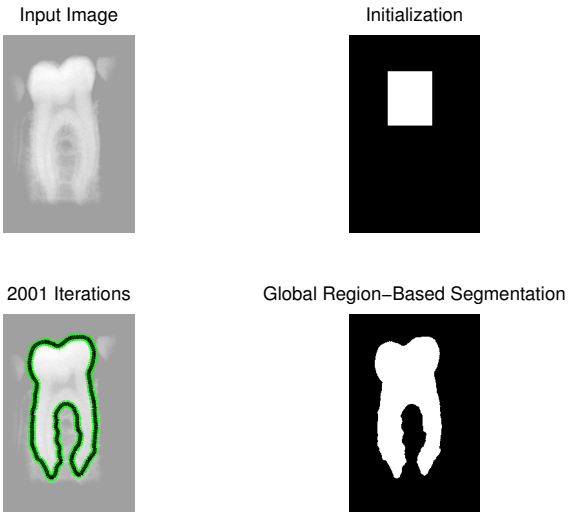

Figure 18: With blackening

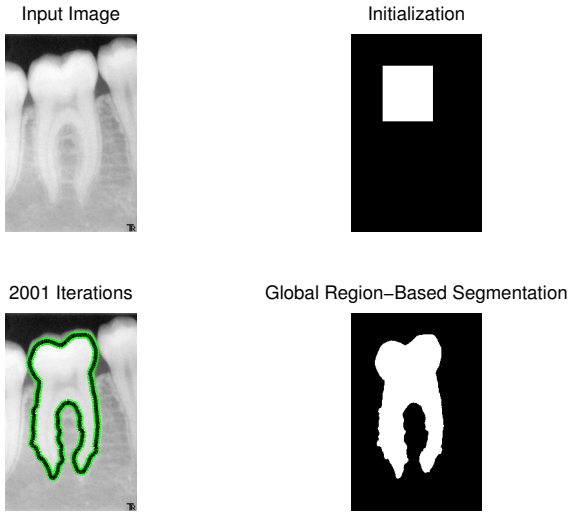

Figure 19: Without blackening

The other pretreament is to blurring the picture, in order to decrease the influence of gum's nerves. Figure (20) show the result of Chan-Vese on blurred rx2 picture, with $\alpha = 0$. Blur the allows to get ripped of the smooth term. It actually provide a better clustering than Otsu/Clustering methods as it tends to exihibit geometrics "blocks". Though, we still need to initialise the Chan-Vese algorithm on blurred image.

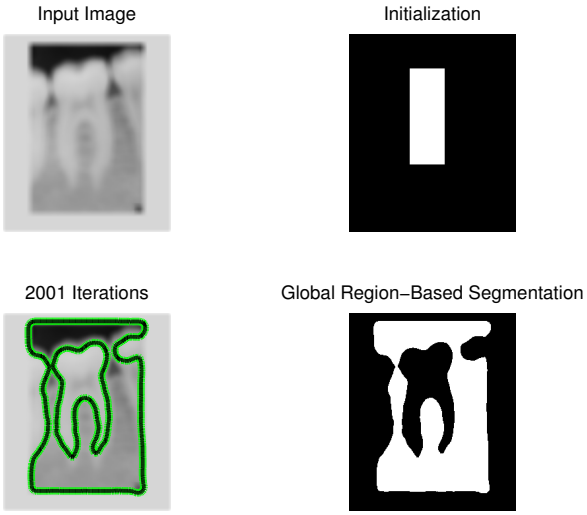

Figure 20: Chan-Vese on a blur image, $\alpha = 0$

4 Appendix

4.1 Local Chan-Vese

Local Chan-Vese code

```
% Localized Region Based Active Contour Segmentation:
%
% seg = localized_seg(I,init_mask,max_its,rad,alpha,method)
%
% Inputs: I           2D image
%         init_mask   Initialization (1 = foreground, 0 = bg)
%         max_its     Number of iterations to run segmentation for
%         rad         (optional) Localization Radius (in pixels)
%                       smaller = more local, bigger = more global
%         alpha       (optional)  Weight of smoothing term
%                       higer = smoother
%         method      (optional) selects localized energy
%                       1 = Chan-Vese Energy
%                       2 = Yezzi Energy (usually works better)
%
% Outputs: seg        Final segmentation mask (1=fg, 0=bg)
%
%
% Coded by: Shawn Lankton (www.shawnlankton.com)
%-------------------------------------------------------------------------

function seg = localized_seg(I,init_mask,max_its,rad,alpha,method,display)

  %-- default value for parameter alpha is .1
  if(~exist('alpha','var'))
    alpha = 0.1;
  end
  %-- default value for parameter method is 2
  if(~exist('method','var'))
    method = 2;
  end
  %-- default behavior is to display intermediate outputs
  if(~exist('display','var'))
    display = true;
  end
  %-- Ensures image is 2D double matrix
  I = im2graydouble(I);
  %-- Default localization radius is 1/10 of average length
  [dimy dimx] = size(I);
  if(~exist('rad','var'))
    rad = round((dimy+dimx)/(2*8));
```

```
  if(display>0)
    disp(['localiztion radius is: ' num2str(rad) ' pixels']);
  end
end

%-- Create a signed distance map (SDF) from mask
phi = mask2phi(init_mask);

%--Init

its = 1;
%--main loop
while its <= max_its  % Note: no automatic convergence test

  %-- get the curve's narrow band
  idx = find(phi <= 1.2 & phi >= -1.2)';
  [y x] = ind2sub(size(phi),idx);

  %-- get windows for localized statistics
  xneg = x-rad; xpos = x+rad;      %get subscripts for local regions
  yneg = y-rad; ypos = y+rad;
  xneg(xneg<1)=1; yneg(yneg<1)=1;  %check bounds
  xpos(xpos>dimx)=dimx; ypos(ypos>dimy)=dimy;

  %-- re-initialize u,v,Ain,Aout
  u=zeros(size(idx)); v=zeros(size(idx));
  Ain=zeros(size(idx)); Aout=zeros(size(idx));

  %-- compute local stats
  for i = 1:numel(idx)  % for every point in the narrow band
    img = I(yneg(i):ypos(i),xneg(i):xpos(i)); %sub image
    P = phi(yneg(i):ypos(i),xneg(i):xpos(i)); %sub phi

    upts = find(P<=0);             %local interior
    Ain(i) = length(upts)+eps;
    u(i) = sum(img(upts))/Ain(i);

    vpts = find(P>0);              %local exterior
    Aout(i) = length(vpts)+eps;
    v(i) = sum(img(vpts))/Aout(i);
  end

  %-- get image-based forces
  switch method  %-choose which energy is localized
   case 1,                   %-- CHAN VESE
    F = -(u-v).*(2.*I(idx)-u-v);
   otherwise,                %-- YEZZI
    F = -((u-v).*((I(idx)-u)./Ain+(I(idx)-v)./Aout));
  end
```

```
    %-- get forces from curvature penalty
    curvature = get_curvature(phi,idx,x,y);

    %-- gradient descent to minimize energy
    dphidt = F./max(abs(F)) + alpha*curvature;

    %-- maintain the CFL condition
    dt = 2/(max(dphidt)+eps);

    %-- evolve the curve
    phi(idx) = phi(idx) + dt.*dphidt;

    %-- Keep SDF smooth
    phi = sussman(phi, .5);

    %-- intermediate output
    if((display>0)&&(mod(its,20) == 0))
      showCurveAndPhi(I,phi,its);
    end
    its=its+1;
  end

  %-- final output
  if(display)
    showCurveAndPhi(I,phi,its);
  end
  figure;
  plot(ITS,DELT_PHI);
  %-- make mask from SDF
  seg = phi<=0; %-- Get mask from levelset

%---------------------------------------------------------------------
%---------------------------------------------------------------------
%-- AUXILIARY FUNCTIONS ----------------------------------------------
%---------------------------------------------------------------------
%---------------------------------------------------------------------

%-- Stopping criterion
function stop = isvarlow(DELT_PHI, ITS, tol)

    nb_its = ITS(length(ITS));

    if nb_its < 20
        stop = false;
    else
        % Calcul de la  d riv
        D = diff(DELT_PHI(2:nb_its));
        % Elimination du bruit
```

```
        [b,a]=butter(2,0.02,'low');
        y=filter(b,a,D);
        % Critre d'arrt : sur les 20 dernires itrations N(dphi) est rest
        % autour de 0.
        stop = ((abs((y(nb_its-19:nb_its-2))) > tol)*ones(18,1)) < 0.5;
    end
%-- Displays the image with curve superimposed
function showCurveAndPhi(I, phi, i)
  imshow(I,'displayrange',[0 255]); hold on;
  contour(phi, [0 0], 'g','LineWidth',4);
  contour(phi, [0 0], 'k','LineWidth',2);
  hold off; title([num2str(i) ' Iterations']); drawnow;

%-- converts a mask to a SDF
function phi = mask2phi(init_a)
  phi=bwdist(init_a)-bwdist(1-init_a)+im2double(init_a)-.5;

%-- compute curvature along SDF
function curvature = get_curvature(phi,idx,x,y)
    [dimy, dimx] = size(phi);

    %-- get subscripts of neighbors
    ym1 = y-1; xm1 = x-1; yp1 = y+1; xp1 = x+1;

    %-- bounds checking
    ym1(ym1<1) = 1; xm1(xm1<1) = 1;
    yp1(yp1>dimy)=dimy; xp1(xp1>dimx) = dimx;

    %-- get indexes for 8 neighbors
    idup = sub2ind(size(phi),yp1,x);
    iddn = sub2ind(size(phi),ym1,x);
    idlt = sub2ind(size(phi),y,xm1);
    idrt = sub2ind(size(phi),y,xp1);
    idul = sub2ind(size(phi),yp1,xm1);
    idur = sub2ind(size(phi),yp1,xp1);
    iddl = sub2ind(size(phi),ym1,xm1);
    iddr = sub2ind(size(phi),ym1,xp1);

    %-- get central derivatives of SDF at x,y
    phi_x  = -phi(idlt)+phi(idrt);
    phi_y  = -phi(iddn)+phi(idup);
    phi_xx = phi(idlt)-2*phi(idx)+phi(idrt);
    phi_yy = phi(iddn)-2*phi(idx)+phi(idup);
    phi_xy = -0.25*phi(iddl)-0.25*phi(idur)...
             +0.25*phi(iddr)+0.25*phi(idul);
    phi_x2 = phi_x.^2;
    phi_y2 = phi_y.^2;

    %-- compute curvature (Kappa)
```

```
    curvature = ((phi_x2.*phi_yy + phi_y2.*phi_xx - 2*phi_x.*phi_y.*phi_xy)./...
                (phi_x2 + phi_y2 +eps).^(3/2)).*(phi_x2 + phi_y2).^(1/2);

%-- Converts image to one channel (grayscale) double
function img = im2graydouble(img)
  [dimy, dimx, c] = size(img);
  if(isfloat(img)) % image is a double
    if(c==3)
      img = rgb2gray(uint8(img));
    end
  else           % image is a int
    if(c==3)
      img = rgb2gray(img);
    end
    img = double(img);
  end

%-- level set re-initialization by the sussman method
function D = sussman(D, dt)
  % forward/backward differences
  a = D - shiftR(D); % backward
  b = shiftL(D) - D; % forward
  c = D - shiftD(D); % backward
  d = shiftU(D) - D; % forward

  a_p = a;  a_n = a; % a+ and a-
  b_p = b;  b_n = b;
  c_p = c;  c_n = c;
  d_p = d;  d_n = d;

  a_p(a < 0) = 0;
  a_n(a > 0) = 0;
  b_p(b < 0) = 0;
  b_n(b > 0) = 0;
  c_p(c < 0) = 0;
  c_n(c > 0) = 0;
  d_p(d < 0) = 0;
  d_n(d > 0) = 0;

  dD = zeros(size(D));
  D_neg_ind = find(D < 0);
  D_pos_ind = find(D > 0);
  dD(D_pos_ind) = sqrt(max(a_p(D_pos_ind).^2, b_n(D_pos_ind).^2) ...
                       + max(c_p(D_pos_ind).^2, d_n(D_pos_ind).^2)) - 1;
  dD(D_neg_ind) = sqrt(max(a_n(D_neg_ind).^2, b_p(D_neg_ind).^2) ...
                       + max(c_n(D_neg_ind).^2, d_p(D_neg_ind).^2)) - 1;

  D = D - dt .* sussman_sign(D) .* dD;
```

```
%-- whole matrix derivatives
function shift = shiftD(M)
  shift = shiftR(M')';

function shift = shiftL(M)
  shift = [ M(:,2:size(M,2)) M(:,size(M,2)) ];

function shift = shiftR(M)
  shift = [ M(:,1) M(:,1:size(M,2)-1) ];

function shift = shiftU(M)
  shift = shiftL(M')';

function S = sussman_sign(D)
  S = D ./ sqrt(D.^2 + 1);
```

Demo of the Local Chan-Vese method

To run the local active contour method demo, write `localized_seg_demo` in the `MatLab` prompt. You can set different parameters such as the mask initialisation and local method parameters. Here is the code :

```
close all;

I = imread('rx2.jpg');  %-- load the image
I = imresize(I,.5);  %-- make image smaller
subplot(2,2,1); imshow(I); title('Input Image');

%-- Choose the mask initialisation : rectangular or ellipse
%-- Create initial mask (ellipse selection)
[xa,ya]=ginput(2);
[xb,yb]=ginput(2);
Nb = 10000;
C = 'b';
me=ellipse(xa,ya,xb,yb,C,Nb,I);
[xa,ya]=ginput(2);
[xb,yb]=ginput(2);
Nb = 1000;
C = 'm';
mi=ellipse(xa,ya,xb,yb,C,Nb,I);
mi=abs(mi-1); %Complementaire
id=(find(mi==1&me==1));
m=zeros(size(I,1),size(I,2));
m(id)=1;

% %-- Create initial mask (rectangular selection)
% m = select_init_mask(I);
```

```
subplot(2,2,2); imshow(m); title('Initialization');
subplot(2,2,3); title('Segmentation');

%-- Segmentation parameter
max_its=4000;
rad=floor(size(I,2)/15);
alpha=0.8;
method=1;
display=true;

%-- Local circle for the segmentation
subplot(2,2,1);
theta = 0:0.001:2*pi;
x0 = size(I,2)-rad-1;
y0 = rad+1;
hold on;
plot(rad*cos(theta)+x0,rad*sin(theta)+y0,'r-');

%-- Segmentation
subplot(2,2,3);
seg = localized_seg(I,m,max_its,rad,alpha,method,display);

%-- Plot
figure(1);
subplot(2,2,4); imshow(seg); title('Global Region-Based Segmentation');
```

4.2 Clustering

im2mask

The first code is **im2mask.m**, which creates a mask for Chan-Vese initialisation from an image and a method as described in the code documentation. It uses the **GKmeans.m** file and **sqdist.m**. You can modify several parameters inside the code of **im2mask** such as the method used for the clustering (Global Kmeans or MatLab Kmeans), or the number of cluster.

```
function M = im2mask(I,method)
% INPUTS :
% I : initial image
% method : 1 -> Global Kmeans
%          2 -> kmeans matlab
% OUPUTS :
% mask : mask (format double) used in the initialisation of Chan Vese algorithm

I = double(I(:,:,1));
k=3; %Kmeans parameter
```

```
points = I(:);
if (method == 1)
    dist=2; %Dist used in GKmeans
    %- Initialisation of Kmeans with Otsu centroid
    IDX=otsu(I,k);
    centre=zeros(k,1);
    for i=1:k
        ind = find(IDX==i);
        centre(i)=mean(points(ind));
    end
    %- Kmeans
    [Er,C,iter] = GKmeans(points, [], k, dist, centre);

    %- Display
    disp('##########################');
    disp('Initial centroid');
    centre
    disp('##########################');
    disp('Final centroid');
    C
    disp('##########################');
    disp('Iter');
    iter
    distance = sqdist(C',points',dist);
    [~,M] = min(distance,[],1);
else
    %- Kmeans
    [M,C,sumd]=kmeans(points,k,'start','uniform','distance','cityblock');
%     [M,C,sumd]=kmeans(points,k,'start','uniform');

    %- Display
    disp('##########################');
    disp('Centroid');
    C
    disp('##########################');
    disp('sums of point-to-centroid distances');
    sumd
end
M = reshape(M,size(I,1),size(I,2));
%- M is a size(I,1)*size(I,2) matrix; M(i,j) == c if point of coordinates
%(i,j) belongs to cluster c; c in [1,k];

%- Search the cluster id of the tooth (k=3 -> 3 cluster thus 1 id
% k=4 -> 4 clusters thus 2 id
Caux = C;
idTooth = [];
for i=1:(k-2)
    [~,id] = max(Caux);
    Caux(id) = 0;
```

```
    idTooth=[idTooth,id];
end
%- Mask creation for Chan-Vese algorithm
for i=1:k
    if (ismember(i,idTooth))
        M(find(M==i))=1;
    else
        M(find(M==i))=0;
    end
%         M(find(M==i))=C(i);
end
```

TestAuto

`TestAuto.m` runs an automatic test from an image I and gives the extracted tooth from the Chan-Vese method.

```
close all;
%- Mask creation
I = imread('rx4.jpg');
m = im2mask(I,2);
% Mask smoothing, sometimes usefull
%m = mask_smoothing(m);

%- Chan-Vese Segmentation
I = imresize(I,.5);  %-- make image smaller
m = imresize(m,.5);
subplot(2,2,1); imshow(I); title('Input Image');
subplot(2,2,2); imshow(m); title('Initialization');
subplot(2,2,3); title('Segmentation');

%-- Segmentation parameters
max_its=4000;
tol=0;
rad=floor(size(I,2)/15);
alpha=0.8;
method=2;
display=true;

%-- Local circle for the segmentation
subplot(2,2,1);
theta = 0:0.001:2*pi;
x0 = size(I,2)-rad-1;
y0 = rad+1;
hold on;
plot(rad*cos(theta)+x0,rad*sin(theta)+y0,'r-');
```

```
%-- Segmentation (local or global)
subplot(2,2,3);
seg = localized_seg(I,m,max_its,rad,alpha,method,display);
% seg = region_seg(I,m,max_its,alpha,display);


%-- Plot
figure(1);
subplot(2,2,4); imshow(seg); title('Global Region-Based Segmentation');
```

4.3 Principal Component Analysis

recText

`recText` aims at finding the gum in a dental radiography. Its inputs are an image I, and the path to the gum basis matrix and the other parts of the tooth basis matrix. The matrix columns are the vector's bloc basis from the PCA, and the last row is the mean from the set of points used in the PCA. There are already two database (so four matrix) in our Dropbox files : one was created by using three textures on each radiography, the other by using six textures. `recText` output is a black and white mask (the gum is black, other parts are white). Here is the code :

```
function T = recText(I,dirG,dirA)
%% Doc
% INPUTS : I : dental radiography image
% dirG and dirA are paths to the gum and other part database matrix
% OUPUTS : T : black and white mask (the gum is black, other parts are white).
%% Initilisation
I = I(:,:,1);
[ni,mi] = size(I);
T = zeros(ni,mi);
load(dirA);
load(dirG);
nTextG = size(G,1);
nTextA = size(A,1);
w = sqrt(size(G{1},2));
nText = nTextA + nTextG;
%% Reconnaissance Texture
IB=split_into_blocs(I,ones(w,w));
p = size(IB,3);
IBC=zeros(w^2,p);
for j=1:p
    tmp=IB(:,:,j);
    IBC(:,j)=tmp(:);
end
```

```
Proj=zeros(nText,p);
coeffProj=3;
for t=1:nTextG
    tmp=G{t};
    mean=tmp(w^2+1,:);
    B=tmp(1:w^2,:);
    tmp=IBC-repmat(mean',1,p);
    Proj(t,:)=sum(abs(B(:,1:coeffProj)'*tmp),1); %k*p
end
for t=(nTextG+1):(nText)
    tmp=A{t-nTextG};
    mean=tmp(w^2+1,:);
    B=tmp(1:w^2,:);
    tmp=IBC-repmat(mean',1,p);
    Proj(t,:)=sum(abs(B(:,1:coeffProj)'*tmp),1); %k*p
end
[~,text]=min(Proj,[],1);
text=(text<=nTextG);
tmp=ones(w,w,p);
for k=1:p
    tmp(:,:,k)=tmp(:,:,k)*text(k)*255;
end
Iapp=uint8(replace_blocs(tmp,ni,mi));
figure(1);
imshow(Iapp);
T=Iapp;
end
```

TestDent

`TestDent` is a script used to create the gum database. First, we select `nText` regions of texture we want to analyse, then we run the PCA on those regions. We then try to rebuild every bloc of the radiography by projecting those blocs on the `nText` texture basis created before. Finally, we try to sort each bloc by their texture by using the two criteria described in the PCA section (each texture has its own intensity of gray). In the code, we can modify the number of texture (`nText`), or the size of a bloc (`w`) used in the PCA, or the number of basis vector used in the picture reconstruction (`coeffProj`). Here is the code :

```
close all;
% load('ImageCouleur.mat');
I=imread('rx1.bmp');
I=double(I(:,:,1));
%I=rgb2gray(I);
disp('nombre texture :');
```

```
nText=3
nFig=1;
figure(nFig);
imshow(I);
[ni,mi]=size(I);
disp('################ Partie 1 ################');
%%
disp('############ Selection des regions ################');
rSR=zeros(4,nText);
for t=1:nText
    rSR(:,t) = getrect(figure(1))';
end
%%
disp('############ Split block et PCA ################');
disp('valeur de w:');
w=2
aux=ones(w,w);
for t=1:nText
    xmin=rSR(1,t);
    ymin=rSR(2,t);
    width=rSR(3,t);
    height=rSR(4,t);
    SPB=split_into_blocs(I(ymin:(ymin+height),xmin:(xmin+width)),aux);
    p = size(SPB,3);
    X=zeros(p,w^2);
    for j=1:p
        tmp=SPB(:,:,j);
        X(j,:)=tmp(:)';
    end
    m = mean(X);
    XC = X - repmat(m,p,1);
    [B,D] = eig(XC'*XC/w^2);
    [D, ind] = sort(diag(D), 1, 'descend');
    B = B(:,ind);
    name=strcat('B',num2str(t));
    assignin('base',name,B);
    name=strcat('D',num2str(t));
    assignin('base',name,D);
    name=strcat('mean',num2str(t));
    assignin('base',name,m);
end
%%
disp('################ Partie 2 ################');
disp('valeur de k:');
k=2
disp('ratio eigenvalue:')
for t=1:nText
    tmp=evalin('base',strcat('D',num2str(t)));
    sum(tmp(1:k))/sum(tmp)
```

```
end
for t=1:nText
    IB=split_into_blocs(I,ones(w,w));
    p = size(IB,3);
    mean=evalin('base',strcat('mean',num2str(t)));
    B=evalin('base',strcat('B',num2str(t)));
    IBC=zeros(w^2,p);
    for j=1:p
        tmp=IB(:,:,j);
        IBC(:,j)=tmp(:);
    end
    IBC=IBC-repmat(mean',1,p);
    IBC=B(:,1:k)*B(:,1:k)'*IBC+repmat(mean',1,p); %w^2*p
    tmp=zeros(w,w,p);
    for j=1:p
        tmp(:,:,j)=reshape(IBC(:,j),w,w);
    end
    name=strcat('IBmod',num2str(t));
    assignin('base',name,tmp);
    IBmod1=tmp;
    Imod=uint8(replace_blocs(tmp,ni,mi));
    nFig=nFig+1;
    figure(nFig);
    imshow(Imod);
end

%% Appartenance a une texture

disp('############ Appartenance ################');

%%
disp('################# Erreur coefficient projection ################');

IB=split_into_blocs(I,ones(w,w));
p = size(IB,3);
IBC=zeros(w^2,p);
for j=1:p
    tmp=IB(:,:,j);
    IBC(:,j)=tmp(:);
end
Proj=zeros(nText,p);
coeffProj=1;
for t=1:nText
    mean=evalin('base',strcat('mean',num2str(t)));
    B=evalin('base',strcat('B',num2str(t)));
    tmp=IBC-repmat(mean',1,p);
    Proj(t,:)=sum(abs(B(:,1:coeffProj)'*tmp),1); %k*p
end
[~,text]=min(Proj,[],1);
```

```
tmp=ones(w,w,p);
for k=1:p
    tmp(:,:,k)=tmp(:,:,k)*text(k)*255/nText;
end
Iapp=uint8(replace_blocs(tmp,ni,mi));
nFig=nFig+1;
figure(nFig);
imshow(Iapp);

%%
disp('################# Erreur norme L2 ################');
IB=split_into_blocs(I,ones(w,w));
p=size(IB,3);
err=zeros(3,p);
for t=1:nText
    IBmod=evalin('base',strcat('IBmod',num2str(t)));
    er=IB-IBmod;
    for k=1:p
        tmp=er(:,:,k);
        err(t,k)=norm(tmp);
    end
end
tmp=ones(w,w,p);
for k=1:p
    [~,ind]=min(err(:,k));
    tmp(:,:,k)=tmp(:,:,k)*ind*255/nText;
end
Iapp=uint8(replace_blocs(tmp,ni,mi));
figure;
nFig=nFig+1;
figure(nFig);
imshow(Iapp);
```

5 Bibliography

References

[1] Shawn M.Lankton, *"Localised statistical models in computer vision"*, PHD Thesis, Georgia Institute of Technology, 2009.

[2] Ross Whitaker, *"A level-set approach to 3d reconstruction from range data."*, International Journal of Computer Vision, R. T. 1998.

[3] Shawn M.Lankton, *"Sparse Field Methods - Technical Report"*, Technical Report, 2009.

[4] Aristidis Likas, Nikos Vlassis, Jakob J. Verbeek, *"The global k-means clustering algorithm"*, The journal of the pattern recognition society, 2009.