# LUNG CAD SYSTEM – 5 11 2007

## *Documentazione Generale*

### Introduzione

Il documento descrive, nella prima parte (Lung CAD System), il funzionamento logico del CAD.
La seconda parte contiene la documentazione relativa all'implementazione software del sistema.

## Indice generale

# LUNG CAD SYSTEM – 5 11 2007

## *System overview:*

The CAD system comprises at present four macro-steps:

1. Pre-processing:
    a. CT stack resizing (present: z-axis; future: x, y, z);
    b. Lung segmentation;
2. 2D nodule-like signals detection in slices: logical *AND* of
    a. Fast Radial filter;
    b. Scale Space filter;
3. Grouping of 2D signals into 3D candidate nodules;
4. False Positive Reduction (FPR):
    a. Coarse FPR (on the basis of a few simple features);
    b. Fine FPR:
        i. 2D Gray Level features and Support Vector Machine classifier;
        ii. 3D Ranklet-Based features and Support Vector Machine classifier;
        iii. 2D Support Vector Regression Filtering FPR.

    Steps i, ii, iii can be combined in different modalities.

*Note: matlab parameters have been added to facilitate readability when matlab – c bridging.*

## 1) Pre-processing

### CT stack resizing

The system requires CT exams with slice thickness equal to slice spacing (no overlapping), and be set before processing. In order to reduce the number of parameters of the algorithm, we decided that every patient stack must undergo a resampling operation, in case their spacing and thickness values are different from the required ones, hence the need of a z-axis linear resizing. As a future development, tri-linear resizing will be introduced, with the aim to perform the detection step with 3D filters (which need isotropic voxels).

### Lung Segmentation

Our segmentation algorithm is composed of six main steps: i) a smoothing algorithm is applied to the CT stack to reduce noise; ii) the lung region is extracted from the CT images by adaptive grey-level thresholding; iii) trachea region is eliminated from initial CT slices; iv) the left and right lungs are then separated to permit finer processing on each lung separately; v) some spurious regions, left over from the segmentation step based on region volume size, are eliminated; vi) lung contour is smoothed, in order to retain lung structures within the lung and include nodules attached to the lung wall. The overall segmentation process is described in Fig. 1, together with the type of data involved in each processing step.

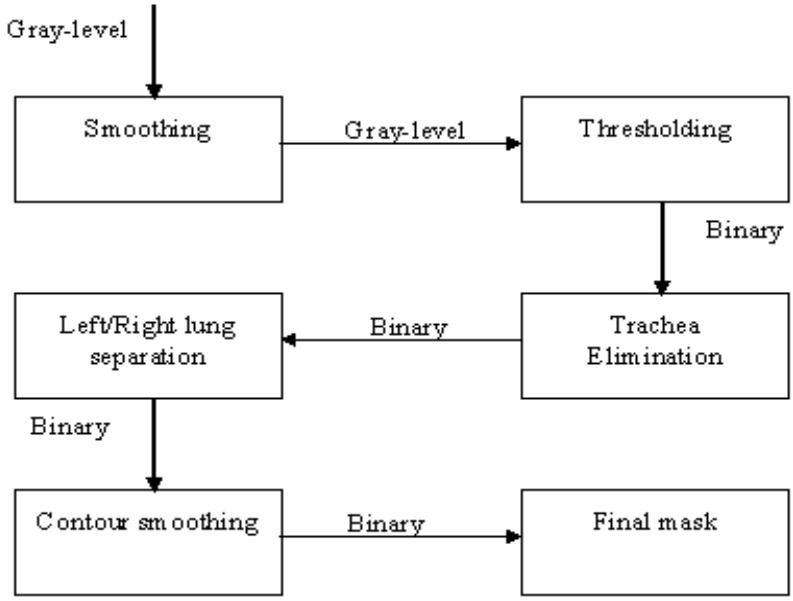



**Figure 1: Overall segmentation algorithm. Arrows show direction of data flow.**

Matlab parameters:

```
%z-axis resized patients (1) or just original ones (0)
RESIZED_PAT = 1;
%thickness and spacing
SETSPACING = 2.0;
SETTHICKNESS = 2.0;
LITTLE_TOLERANCE = 0.0001;
SIZELIMIT  = 4.9;%minimum diameter(mm) of nodules considered for final statistics
```

## 2) 2D nodule-like signals detection in slices

Each slice in the segmented CT stack is processed in the following way:

i. Fast Radial filtering [Loy03]: it is a sort of non-linear and shape dependent Gray Level transformation. It enhances circular bright spots more than non-circular ones, hence it shows high specificity to nodule-like signals. This image undergoes a second identical
ii. Fast Radial filtering. Filtered image is then thresholded to find high peaks, corresponding to detected signals. Threshold is optimized with a Cross Validation procedure.
iii. All these signals are subsequently considered for the analysis with Scale Space techniques. According to Lindeberg [Lindeberg93] Scale Space extrema - points that are local extrema both in scale and space - of the Normalized Laplacian reflect a characteristic length of the objects in the image. Moreover, it has been found that there is a direct relationship between the sigma of the Normalized Laplacian employed for filtering and the radius of circular or quasi-circular bright objects (*sigma resonance*: $\sigma = \frac{r}{\sqrt{2}}$ ). By using this technique, it is possible to easily estimate the size of the detected objects, and to eliminate all those signals whose dimensions are out of the predefined detection range (usually, 4 to 20 mm diameter), without the need to determine the exact borders of the signals.

    The last version of the algorithm slightly differs from the first one: because many different resonance points can be found for a single FR signal, instead of simply calculating the mean value of the sigma values (and a sigma-weighted sum for the coordinates), we consider the

bounding box of all the found signals (for a given FR signal), and define it to be the radius, if it is not too much large, compared with each single signal (...).

The logical AND with the Scale Space filter reduces the number of False Positives detected by the FR filtering step to 80÷90% of the initial value.

Matlab parameters:

```
%fast radial parameters
FR.RADII = [3 5 7 9];
FR.FRPAR = 2;%how circular?...
FR.THR = 1.45%thresholding value - test
FR.THRSVM = 0.95;%threshold value for svm model: used in training
FR.RATIOTHR = 1.85;% 01 10 2007, new feature algorithm: minsigma times threshold
FR.SIGMATHR = 1.19;%threhsold over sigma in 3D objects 03 10 2007
FR.QFACTOR = 1.4;% quality factor for bounding box

FR.min_nod_diam = 3.8%mm: minimum diameter for LESS
FR.max_nod_diam = 22.0;%mm: maximum diameter for LESS

%choose sigma for ROI side calculation
FPRPARAM.WHICHSIGMA = 9; %5: max(sigma); 6: min(sigma); 8: mean(sigma); 9: new
algorithm sigma @ 01 10 2007
FPRPARAM.CNTRD_TOL = 6; %tolerance for nodules centroid when comparing to GT
```

## 3) Grouping of 2D signals into 3D candidate nodules

After setting a proper spatial tolerance, 2D signals are matched across slices by simply comparing their positions, determined by means of the Scale Space procedure. Beginning from the first signal in the first slice, each signal is linked with each signal in the next slice, provided their spatial positions are within the chosen tolerance. The result of this operation is an ensemble of groups of 2D signals corresponding to objects extended across slices in the CT scan of patients: these objects are the candidate nodules that undergo a False Positive Reduction step.

Matlab parameters:

```
FPRPARAM.DELTA_TOL = 7.0; %COORDINATE TOLERANCE FOR OBJECTS ACROSS SCANS
%objects across scans are linked together if their gap is 0 or 1
FPRPARAM.MAXSCANDISTANCE = 3;  %<-> gap = 1; if 2, gap = 0;
```

## 4) False Positive Reduction

**Coarse FPR**

It is possible to cut all signals which are too short or too long, those which are too much inclined with respect to the z-axis (nodules are typically not inclined, vessels are very much inclined), and those whose volume is too large. In particular, too short means the signal is a singleton (it is linked with no other signals) and too long is related to the maximum size of searched nodules.
Another parameter has recently been introduced: there is a maximum allowed value for the average grey level of each candidate nodule.
Approximately 70% to 80% of false nodules are eliminated by this FPR step, and at the same time only 5÷10% of nodules are lost.

Matlab parameters:

```
% parameters used in training
% 0.95 res 3.0to2.0; 2.8 --> 22.0
FPRPARAM.MAXTHETA.L2 = 72;
FPRPARAM.MAXTHETA.L3 = 68;
FPRPARAM.MAXTHETA.L4 = 67;
FPRPARAM.MAXTHETA.L5 = 67;
FPRPARAM.MAXTHETA.L6 = 39;
FPRPARAM.MAXTHETA.L7 = 39;
FPRPARAM.MAXTHETA.L8 = 32;
FPRPARAM.MAXTHETA.L9 = 32;
FPRPARAM.MAXTHETA.L10 = 15.0;

FPRPARAM.MAXLEN = 13;

FPRPARAM.MAXVOL.L2 = 0;
FPRPARAM.MAXVOL.L3 = 0;
FPRPARAM.MAXVOL.L4 = 8.5;
FPRPARAM.MAXVOL.L5 = 17.5;
FPRPARAM.MAXVOL.L6 = 35;
FPRPARAM.MAXVOL.L7 = 48;
FPRPARAM.MAXVOL.L8 = 81;
FPRPARAM.MAXVOL.L9 = 81;
FPRPARAM.MAXVOL.L10 = 85;
FPRPARAM.MAXVOL.LRIGHT2 = 42;
FPRPARAM.MAXVOL.LRIGHT3 = 83;
FPRPARAM.MAXVOL.LRIGHT4 = 93;
FPRPARAM.MAXVOL.LRIGHT5 = 103;
FPRPARAM.MAXVOL.LRIGHT6 = 112;
FPRPARAM.MAXVOL.LRIGHT7 = 152;
FPRPARAM.MAXVOL.LRIGHT8 = 175;
FPRPARAM.MAXVOL.LRIGHT9 = 200;
FPRPARAM.MAXVOL.LRIGHT10 = 225;
FPRPARAM.MAXVOL.LRIGHT = 225;

%parameters to be used in test
FPRPARAM.MAXTHETA_T.L2 = 72;
FPRPARAM.MAXTHETA_T.L3 = 68;
FPRPARAM.MAXTHETA_T.L4 = 67;
FPRPARAM.MAXTHETA_T.L5 = 67;
FPRPARAM.MAXTHETA_T.L6 = 39;
FPRPARAM.MAXTHETA_T.L7 = 39;
FPRPARAM.MAXTHETA_T.L8 = 32;
FPRPARAM.MAXTHETA_T.L9 = 32;
FPRPARAM.MAXTHETA_T.L10 = 15.0;

FPRPARAM.MAXLEN_T = 13;

FPRPARAM.MAXVOL_T.L2 = 0;
FPRPARAM.MAXVOL_T.L3 = 0;
FPRPARAM.MAXVOL_T.L4 = 8.5;
FPRPARAM.MAXVOL_T.L5 = 17.5;
FPRPARAM.MAXVOL_T.L6 = 35;
FPRPARAM.MAXVOL_T.L7 = 48;
FPRPARAM.MAXVOL_T.L8 = 81;
FPRPARAM.MAXVOL_T.L9 = 81;
FPRPARAM.MAXVOL_T.L10 = 85;
FPRPARAM.MAXVOL_T.LRIGHT2 = 42;
FPRPARAM.MAXVOL_T.LRIGHT3 = 83;
FPRPARAM.MAXVOL_T.LRIGHT4 = 93;
FPRPARAM.MAXVOL_T.LRIGHT5 = 103;
```

```
FPRPARAM.MAXVOL_T.LRIGHT6 = 112;
FPRPARAM.MAXVOL_T.LRIGHT7 = 152;
FPRPARAM.MAXVOL_T.LRIGHT8 = 175;
FPRPARAM.MAXVOL_T.LRIGHT9 = 200;
FPRPARAM.MAXVOL_T.LRIGHT10 = 225;
FPRPARAM.MAXVOL_T.LRIGHT = 225;

% training and test
FPRPARAM.MAXGL = 150;  % 05 10 2007: max mean grey level of 3D candidates
```

**Fine FPR**

This step includes three independent branches that can be combined in different ways, according to classifiers combination rules (see, for example, [Kittler98]).
At present only logical AND of point I) and iii) is being tested.

**i) 2D Gray Level features and Support Vector Machine classifier**

*Overview*

This FPR branch comprises two sub-steps: one is the classification of each 2D signal; the other is the final labelling of each group of 2D signals. Eventually, only 3D objects judged as nodules will be prompted to the final CAD user.

*Method: sub-step a*

A training procedure is performed, based on SVM classifier. Square Regions of Interest (ROI) around candidate nodules are selected trough their *sigma resonance* values, without the need to determine their exact borders, and the resized to a common side value. Gray Level features are considered as discriminative features for classification. Each positive training sample is rotated multiple times with the aim to obtain a final classifier with a good degree of rotational invariance and to overcome the problem of small databases at disposal. Multiple rotations (24, 15° each) are considered also at time of classification of each 2D ROI: a percentage of these rotated views is set during a Cross Validation procedure as the minimum number of necessary positive classification to give the ROI a positive label

Matlab parameters:

```
FPRPARAM.MRESIZED = 19;
FPRPARAM.NRESIZED = FPRPARAM.MRESIZED;
%using model trained with 90 deg rotated crops or not
FPRPARAM.ROTATEDTRAIN = 1;
%ENLARGED CROPS TO BE USED FOR MULTIROTATIONS IN TRAINING
FPRPARAM.MULTIROT = 1; %1: yes; 0: no
FPRPARAM.MULTIROTSIDE = sqrt(2);%multiplicative factor for side enlargement
%using 90 deg rotated crops in test
FPRPARAM.ROTATEDTEST = 1;
%24 views for labeling each 2D image, ora just the usual 4 (rotatedtest =
%1) or 1 (rotatedtest = 0)
FPRPARAM.MULTIROTTEST = 1; %1: yes; 0: no
FPRPARAM.MRTTHR = 12;  %minimum number of positive views over the 4 or 24

%choosing positive vectors for training or not
FPRPARAM.CHOOSEVECT = 1;

%choosing a maximum number of FPs in training
FPRPARAM.CHOOSEFP = 1;
TIMESFP = 3;FPs are TIMESFP times 2D Tps in training
```

```
%FPs randomly chosen or not
FPRPARAM.RNDFP = 1;

%parameters for loading SVM model (libsvm)
FPRPARAM.SVM_TYPE = 0;
FPRPARAM.KERNEL_TYPE = 1;
FPRPARAM.DEGREE = 2;
FPRPARAM.GAMMA = 1;
FPRPARAM.COEF0 = 1;
FPRPARAM.COST = 100;
FPRPARAM.NU = 0.5;
FPRPARAM.EPS_SVR = 0.1;
FPRPARAM.PROB_ESTIM = 0;
if FPRPARAM.ROTATEDTRAIN > 0.5
  if FPRPARAM.MULTIROT > 0.5
    if FPRPARAM.CHOOSEVECT > 0.5
            FPRPARAM.WEIGTHP = ones(1,FPRPARAM.NUMFOLD)*TIMESFP;%gap 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            switch TIMESFP
              case 1
                %             FPRPARAM.NUMFP = [6216 5712 6096 5880 6168 6144 6192
            6144 6312 5616];
              case 2
                %             FPRPARAM.NUMFP = [9072 8352 8736 8784 8832 9024 9168
            8784 9120 7824];  %065 2x 2e8 --> 30
                %             FPRPARAM.NUMFP = [8400 7440 7872 7872 8160 8304 8304
            8112 8592 7296];
              case 3
                % FPRPARAM.NUMFP = [8640 9576 9936 9720 10296 10584 8352 9504];
            %075 2x 2e8 --> 30
                  FPRPARAM.NUMFP = [9360 9864 10944 10584 11376 11520 9648 10872];
            %095 2x 2e8 --> 22
              case 4
                  FPRPARAM.NUMFP = [];%075 2x 2e8 --> 30
              case 5

              case 6

              case 7

              otherwise

            end


%         FPRPARAM.WEIGTHP = [12 14 12 14 14 12 14]; %gap 1
%         FPRPARAM.WEIGTHP = [17 29 24 67 67 60 71]; %gap 0
    else
            FPRPARAM.WEIGTHP = [1  1  1  1  1  1  1];%gap 1
      FPRPARAM.NUMFP = [6216 5712 6096 5880 6168 6144 6192 6144 6312 5616];
%         FPRPARAM.WEIGTHP = [17 29 24 67 67 60 71]; %gap 0
    end
  else%multirot
    if FPRPARAM.CHOOSEVECT > 0.5
            FPRPARAM.WEIGTHP = [72  86  75  83  84  70  85];%gap 5
      %         FPRPARAM.WEIGTHP = [17  29  24  67  67  60  71];  %gap 0
    else
```

```
            FPRPARAM.WEIGTHP = [72  86  75  83  84  70  85];  %gap 1
        %       FPRPARAM.WEIGTHP = [17  29  24  67  67  60  71];  %gap 0
      end
   end
else
   FPRPARAM.WEIGTHP = [200.0];
end
FPRPARAM.WEIGHTN = 1.0;

%normalized-data-or-not model
FPRPARAM.NORMALIZE4SVM = 1;
```

*Method: sub-step b*

A heuristic procedure is used, after a Cross Validation optimization step, to give each 3D group of 2D signals (ROIs) a final label: nodule or not-nodule. The positive label is given when a certain percentage of the ROIs of the group are classified as 2D nodules. This percentage depends on the number of ROIs of the group.

Matlab parameters:

```
%considering usual majority voting (0) or modified majority voting (1)
FPRPARAM.MODMAJV = 1; %always!!!!

FPRPARAM.MINMMV = FPRPARAM.MAXLEN_T;    % 13  06  2007
%max length with minimum requirement: at least one positive 2D label
FPRPARAM.HITFORMMV = 1;  %corresponding  minimum hit  number
            % beginning of NOMORE USED
            FPRPARAM.MINMMV2 = 15;  %other lengths with different requirements
            FPRPARAM.HITFORMMV2 = 1;  %correponding requirements
            FPRPARAM.MINMMV3 = 16;
            FPRPARAM.HITFORMMV3 = 6;
            FPRPARAM.MINMMV4 = 17;
            FPRPARAM.HITFORMMV4 = 8;
            FPRPARAM.MINMMV5 = 20;
            FPRPARAM.HITFORMMV5 = 10;
            FPRPARAM.MINMMV6 = 30;
            FPRPARAM.HITFORMMV6 = 12;
            % end of  NOMOREUSED

% new FPR2 heuristic  algorithm :  different  thresholds  for  different  lengths

%simply  the  right  bound on length
FPRPARAM.FPR2LEN.L2  = 2;
FPRPARAM.FPR2LEN.L3  = 3;
FPRPARAM.FPR2LEN.L4  = 4;
FPRPARAM.FPR2LEN.L5  = 5;
FPRPARAM.FPR2LEN.L6  = 6;
FPRPARAM.FPR2LEN.L7  = 7;
FPRPARAM.FPR2LEN.L8  = 8;
FPRPARAM.FPR2LEN.L9  = 9;
FPRPARAM.FPR2LEN.L10  = 10;
FPRPARAM.FPR2LEN.L11  = 11;
FPRPARAM.FPR2LEN.L12  = 12;
FPRPARAM.FPR2LEN.L13  = 13;
FPRPARAM.FPR2LEN.L14  = 14;
FPRPARAM.FPR2LEN.L15  = 15;

%the number of  hits  at the relative  mrtthr
```

```
FPRPARAM.HITFOR.L2  = 2;
FPRPARAM.HITFOR.L3  = 1;
FPRPARAM.HITFOR.L4  = 1;
FPRPARAM.HITFOR.L5  = 1;
FPRPARAM.HITFOR.L6  = 2;
FPRPARAM.HITFOR.L7  = 4;
FPRPARAM.HITFOR.L8  = 5;
FPRPARAM.HITFOR.L9  = 5;
FPRPARAM.HITFOR.L10 = 5;
FPRPARAM.HITFOR.L11 = 5;
FPRPARAM.HITFOR.L12 = 6;
FPRPARAM.HITFOR.L13 = 7;
FPRPARAM.HITFOR.L14 = 6;
FPRPARAM.HITFOR.L15 = 5;

%the relative  mrtthr (over 24 - rotated - views)
FPRPARAM.MRTTHR2.L2  = 13;
FPRPARAM.MRTTHR2.L3  = 13;
FPRPARAM.MRTTHR2.L4  = 14;
FPRPARAM.MRTTHR2.L5  = 14;
FPRPARAM.MRTTHR2.L6  = 14;
FPRPARAM.MRTTHR2.L7  = 13;
FPRPARAM.MRTTHR2.L8  = 14;
FPRPARAM.MRTTHR2.L9  = 14;
FPRPARAM.MRTTHR2.L10 = 14;
FPRPARAM.MRTTHR2.L11 = 14;
FPRPARAM.MRTTHR2.L12 = 14;
FPRPARAM.MRTTHR2.L13 = 14;
FPRPARAM.MRTTHR2.L14 = 12;
FPRPARAM.MRTTHR2.L15 = 11;
```

**ii) 3D Ranklet-Based features and Support Vector Machine classifier**

*Overview*

Contiguous 2D regions of interest found on segmented lung areas from sections of a CT scan are merged to form volumes of interest (VOIs). Feature vectors are then computed by submitting each VOI to the 3D Ranklet transform, a non-parametric orientation-selective and multi-resolution transform [Masotti06]. Finally, a Support Vector Machine (SVM) classifier is used to discriminate VOIs containing nodules from those containing normal tissue.

*Method*

Given a number of VOIs, also known as 3D nodule candidates, feature vectors are calculated by submitting each of them to the 3D ranklet transform. A 2D version of this transform was in fact successfully developed and evaluated by our group in the discrimination of breast tumoral mass from normal tissue.

By submitting VOIs to the 3D ranklet transform, a number of ranklet coefficients are produced. Ranklet coefficients are, for instance, non-parametric. As for the 2D case, in fact, the 3D ranklet transform deals with voxels' ranks rather than with their gray-level intensity values; i.e., given ($v_1$, …,$v_N$) voxels, the intensity value of each $v_i$ is replaced with the value of its order among all the other voxels. Secondly, ranklet coefficients are multi-resolution and orientation-selective. Similarly to the bi-dimensional Haar wavelet transform, in fact, ranklet coefficients can be calculated at different resolutions and orientations (i.e., vertical, horizontal and diagonal) by means of a suitable stretch and shift of the oriented compact supports used for their computation.

As far as classification of the aforementioned feature vectors is concerned, an SVM classifier is adopted.

**iii) 2D Support Vector Regression (SVR) Filtering FPR**

*Overview*
Starting from two well-known facts:

i. SVM-based classification and regression techniques, arisen from Statistical Learning Theory [Vapnik98], have widely demonstrated in recent years their superiority to conventional techniques, such as Multi-Layer Perceptron (MLP);
ii. MLPs are at the basis of the class of image filters known as Neural Filters, to which an interesting FPR method such as MTANN [Suzuki03] belongs;

we developed a modified version of MTANN employing SVR instead of Neural Networks. This SVR filtering approach was initially tested for FPR in a mass detection CAD for Mammography [Angelini05], giving interesting results.
This FPR branch is composed of two sub-steps: the first is the classification of each 2D signal by means of the SVR-based classification technique; the other is the final labelling of each group of 2D signals, similar to the *sub-step b* of point i) of Fine FPR.

*Method: sub-step a*
Each ROI holding a candidate-nodule is filtered with the SVR-based filtering technique: the result is an output image which is subsequently used to determine if the ROI image contains a nodule.
To obtain the filtered image, the SVR filter is applied to sub-regions in each ROI until the whole image is processed: each sub-region is associated by the SVR algorithm to a continuous output value ranging from 0 to 1, representing a measure of the presence of a portion of a nodule in the input sub-region. A weighted sum of the outputs over each image is used to accomplish the FPR task: a threshold is set on the base of a Cross-Validation procedure.

*Method: sub-step b*
It is a heuristic procedure similar to that introduced in point i) of Fine FPR.

Parameters:

```
%single threshold on SVR summed value
SVRTHR = 0.875;
```

## *Conclusion*

The system was initially trained on a small 34 nodule database (slice thickness 5 mm and slice spacing 3 mm), reaching these results: each of the three fine FPR branches alone made the system detect approximately 80% of nodules at 34 FP/Patient, or 65% at 6 FP/Patient, in a Multi-Fold Cross-Validation procedure. It must be noted that nodules in the database are in the range 3 ÷ 10 mm, and that their detectability is negatively affected by slice thickness and spacing, respectively 5 and 3 mm; moreover, the 3D Ranklet-Based FPR method is even more influenced by the relationship between nodule size and slice thickness and by high voxels anisotropy.
In conclusion, taking into account all the disadvantages of our initial database, we consider the system obtained very promising results, and we believe it will strongly improve its performances when trained on a better database, which will also consent to properly validate the FPR combination step, which we highly value and have already successfully tested in our previous works. For "better database" we intend a larger one with proper (fine) slice thickness and spacing. Indeed, at present we are working on the LIDC public database, and we have also begun creating a database of fine CT scans of patients examined at the local University Hospital (slice thickness 1.25 mm and slice spacing 1.25 mm).

## *References*

[Angelini05]: E. Angelini, R. Campanini and A. Riccardi, "Support vector regression filtering for reduction of false positives in a mass detection cad scheme: preliminary result", University of Bologna Preprint, 2005. *Available at: http://amsacta.cib.unibo.it/archive/00000912/*

[Kittler98]: J. Kittler, M. Hatef, R. P. W. Duin and J. Matas, "On Combining Classifiers", *IEEE TPAAMI*, 20, 3:226-239, 1998

[Lindeberg93]: T. Lindeberg, "On Scale Selection for Differential Operators", *Proc.8th Scandinavian Conference on Image Analysis*, May 1993, 857-866

[Loy03]: G. L. Loy and A. Zelinsky, "Fast Radial Symmetry for Detecting Points of Interest", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, 8, 2003

[Masotti06]: M. Masotti, A ranklet-based image representation for mass classification in digital mammograms, *Medical Physics*, 33(10) (2006) 3951-3961

[Suzuki03]: K. Suzuki, S. G. Armato III, F. Li, S. Sone, and K. Doi, "Massive training artificial neural network (MTANN) for reduction of false positives in computerized detection of lung nodules in low-dose computed tomography", *Med. Phys.* 30 (7), 1602-1617, July 2003

[Vapnik98]: V. N. Vapnik, "Statistical Learning Theory", J. Wiley 1998

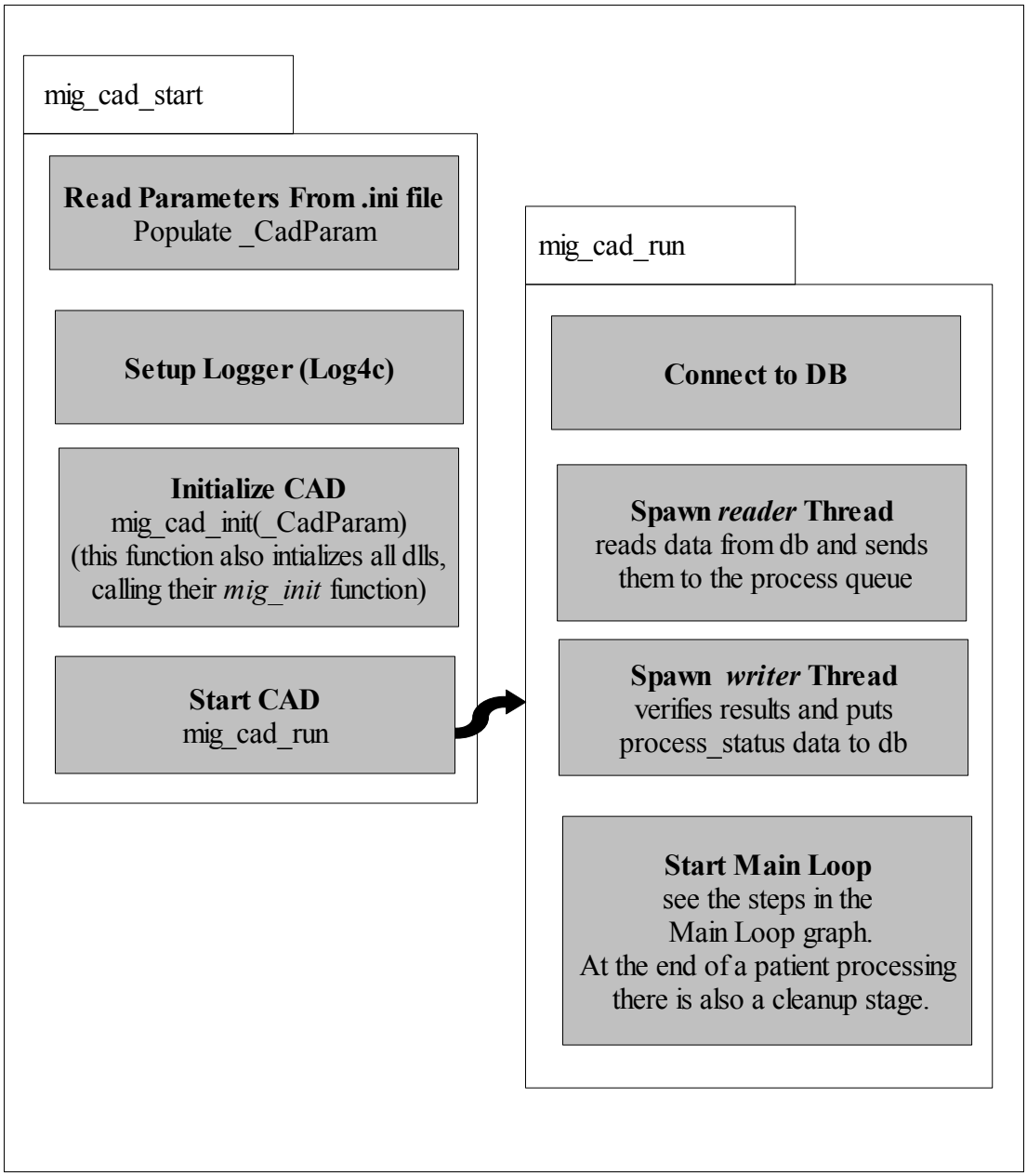
# Struttura Software

# Introduzione

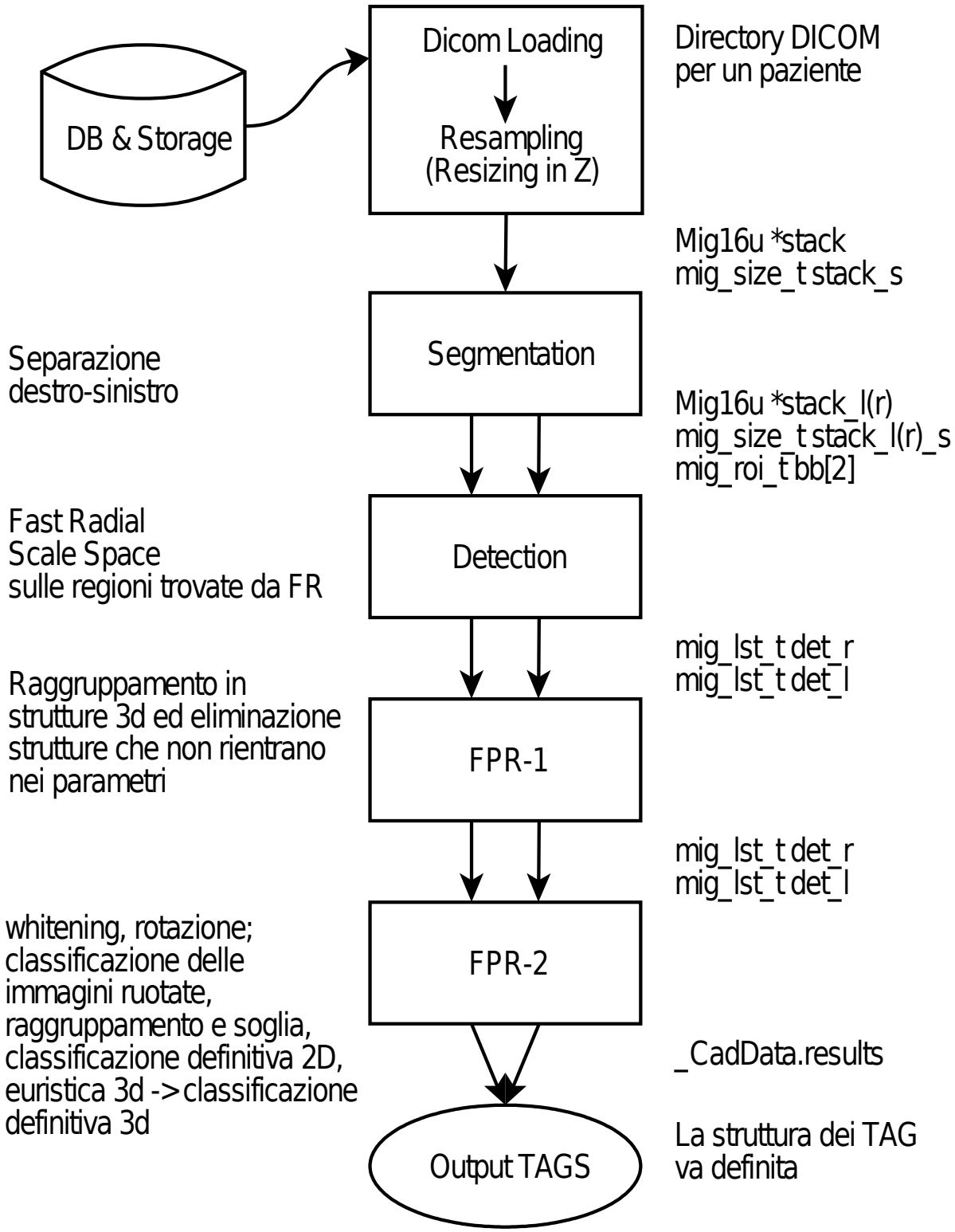
In questa sezione sono raccolte le informazioni relative all'implementazione software del CAD. Sono presentati:

- alcuni diagrammi che rappresentano il flusso del CAD, nel modo in cui è implementato;
- i dettagli della struttura, nella sezione *STRUTTURA SOFTWARE DEL CAD POLMONE*;
- alcuni ulteriori dettagli sugli ingressi e le uscite dei vari blocchi e sulle strutture dati coinvolte;
- un esempio di file di configurazione commentato.

# Main CAD Flow.



# Diagramma a blocchi del loop di elaborazione.




Quando compaiono 2 frecce si intende che l'elaborazione avviene su due *thread* separati per il polmone destro e per quello sinistro. Alla fine i risultati vengono uniti in _CadData.results.

Le varie sezioni di elaborazione del sistema sono compiute da librerie dinamiche caricate

programmaticamente, in base ai nomi di file definiti nei file di configurazione. Nel grafico tutti i blocchi rettangolari rappresentano una dll. Queste librerie hanno tutte la stessa interfaccia, spiegata nella sezione *STRUTTURA SOFTWARE DEL CAD POLMONE*.

Ulteriori dettagli sul flusso dei dati sono presenti nella sezione *Percorso dati INPUT OUTPUT.*

## *STRUTTURA SOFTWARE DEL CAD POLMONE*

In quanto segue BASE_DIR denotera la directory principale dove è stato scaricato il codice sorgente del cad polmone. La struttura su disco delle directories a partire dalla BASE_DIR è la seguente:

**bin** contiene vari eseguibili (windows+linux) per la gestione dei files dicom (dcmtk)e per la gestione del database di pazienti (sqlite).

**Build** contiene i makefiles per linux ed i progetti per visual studio. Vengono qui compilati tutti gli eseguibili e le librerie dinamiche.

**Doc** documentazione + TODO.TXT + questo file.

**Etc** contiene i files di configurazione (.ini) per il cad nonche lo schema sql per il database dei pazienti.

**Extern** contiene gli includes e i binari per le librerie esterne precompilate (visual studio , gcc 32bit, gcc 64bit).

**Libmigdb** codice sorgente libreria comunicazione con il database sqlite.

**libmigdet** codice sorgente libreria dll di detection (obsoleta, non utlizzabile).

**libmigdet_2d** codice sorgente libreria dll di detection 2d (utilizzabile, segue Riccardi).

**libmigdet_3d** codice sorgente libreria dll di detection 3d (da debuggare).

**libmigdicom** codice sorgente libreria dll di caricamento immagini dicom da disco + eventuale resampling dello stack.

**libmigfpr_1** codice sorgente libreria dll fpr1 (utilizzabile, Riccardi).
**libmigfpr_2** codice sorgente libreria dll fpr2 (da finire, Riccardi).

**Libmigim** codice sorgente libreria statica routines di immane processing.

**libmigio** codice sorgente libreria statica routines di IO (mat, tiff).

**Libmigmath** codice sorgente libreria statica routines matematiche.

**Libmigseg** codice sorgente libreria dll di segmentazione.

**libmigst** codice sorgente libreria statica strutture varie (liste, stack, queue, hashtable, etc.).

**libmigsvm** codice sorgente libreria statica wrapper per modello libsvm.

**libmigtag** codice sorgente libreria statica scrittura risultati cad su disco.

**libmigut** codice sorgente libreria statica varie funzioni: timers, bit pack/unpack, caricamento dll dinamico, cpuid, file lock/unlock, gestione files .ini.

**lung_cad** codice sorgente main del cad.

**lung_scp** codice sorgente dicom scp (non utilizzabile).

**lung_tools** codice sorgente C e Matlab per vari strumenti di gestione del cad, esempio: routine per la registrazione del cad come servizio windows.

**mig_config.h** file di configurazione globale del codice cad.

**mig_data_cad.h** definizione della struttura globale mig_cad_data_t che ospita i dati tra i vari passaggi di elaborazione.

**mig_data_dicom.h** definizione della struttura mig_dcm_data_t che contiene i dati dicom relativi al paziente attualmente elaborato.

**mig_data_image.h** definizione della struttura mig_size_t che contiene i dati relativi ad un immagine oppure stack di immagini.

**mig_data_types.h** definizione dei tipi di dati base usati nel cad + valori estremi per questi tipi base.

**mig_defs.h** varie macro + defines.

**mig_error_codes.h** codici errore per il cad (di solito le funzioni ritornano 0 se tutti OK e -1 se c'è stato un errore).

**mig_params_cad.h** le chiavi di hash per recuperare i parametri cad dal file di configurazione .ini + valori di default per alcuni parametri.

**nanni** codice sorgente scritto da Nanni (Ranklet3d etc., da recuperare da svn ed inserire in questa directory).

## Entry point (main).

L'entrata principale del cad si trova nel file BASE_DIR\lungcad\mig_cad_start.c. Vengono letti i parametri del cad dal file .ini (per rileggere tali parametri bisogna riavviare il cad), viene inizializzato il sistema di logging ed il controllo passa alle routines presenti nel file BASE_DIR\lungcad\mig_cad.cpp.

BASE_DIR\lungcad\mig_cad.cpp è il cuore di tutto il cad. Qui vengono caricate le librerie dll necessarie al funzionamento, viene stabilità la connessione con il database dei pazienti e viene gestito il work-flow di tutto il cad. Qui viene definita la struttura singleton globale **"static mig_cad_data_t _CadData"** che serve a passare tutti i parametri da una fase di processing all'altra.

## Struttura globale dati cad.

Campi della struttura globale dati cad insieme all'indicazione di chi riempie quale campo, chi lo usa e chi è responsabile della deallocazione:

typedef struct _mig_cad_data_t
{
**cpuinfo_t cpu** informazioni sul pc su cui sta girando il cad: non utlizzata.

**Riempiti dalla routine di loading (libmigdicom.dll):**

**Mig16u *stack** stack dicom originale oppure resampled, 16bit/pixel, row major order.

**Riempito** dalla dll libmigdicom.
**Serve** a tutte le dll di processing.
**Svuotato** all'interno del mig_cad.cpp alla fine del processing di un paziente.

**mig_size_t stack_s** dimensioni (width,height,slices,resolution) dello stack dicom originale o resampled.

**Riempito** dalla dll libmigdicom.
**Serve** a tutte le dll di processing.

**Svuotato** all'interno del mig_cad.cpp alla fine del processing di un paziente.

**int resampled** flag che indica se lo stack originale sia stato ricampionato ( ricampionare o no viene indicato tramite un parametro all'interno del file dei parametri .ini).

**mig_size_t raw_s** dimensioni (width,height,slices,resolution) dello stack dicom originale prima del ricampionamento.

**Riempito** dalla dll libmigdicom.
**Serve** a tutte le dll di processing.
**Svuotato** all'interno del mig_cad.cpp alla fine del processing di un paziente.

**mig_dcm_data_t dicom_data** dati dicom relativi al paziente in processing.

**Riempito** dalla dll libmigdicom.
**Serve** alla scrittura dei tag finali e alle librerie di comunicazione con il database.
**Svuotato** all'interno del mig_cad.cpp alla fine del processing di un paziente.

**mig_cleanup_f load_cleanup** funzione da usare per ripulire i dati allocati dall routine di loading (libmigdicom.dll).

**Riempito** dalla dll libmigdicom.
**Serve** alla pulizia dello stack dicom originale o ricampionato (chiamato all'interno di mig_cad.cpp).
**Svuotato** non deve essere mai svuotato.

**Riempiti dalla routine di segmentazione (libmigseg.dll):**

**Mig16u *stack_l** stack dicom polmone sinistro segmentato.

**Riempito** dalla dll libmigseg.
**Serve** alla dll di detection.
**Svuotato** all'interno della libreria di detection (libmigdet_2d.dll).

**mig_size_t stack_l_s** dimensioni (width,height,slices,resolution) dello stack dicom polmone sinistro segmentato.

**Riempito** dalla dll libmigseg.
**Serve** alla dll di detection.
**Svuotato** all'interno della libreria di detection (libmigdet_2d.dll).

**Mig16u *stack_r** stack dicom polmone destro segmentato.

**Riempito** dalla dll libmigseg.
**Serve** alla dll di detection.
**Svuotato** all'interno della libreria di detection (libmigdet_2d.dll).

**mig_size_t stack_r_s** dimensioni (width,height,slices,resolution) dello stack dicom polmone destro segmentato.

**Riempito** dalla dll libmigseg.
**Serve** alla dll di detection.
**Svuotato** all'interno della libreria di detection (libmigdet_2d.dll).

**mig_roi_t bb[2]** bounding boxes per il polmone dx [0] e sx [1] segmentati.

**Riempito** dalla dll libmigseg.

**Serve** alla dll di detection.
**Svuotato** all'interno della libreria di detection (libmigdet_2d.dll).

**mig_cleanup_f seg_cleanup** funzione da usare per ripulire i dati allocati dall routine di segmentazione (libmigseg.dll).

**Riempito** dalla dll libmigseg.
**Serve** alla pulizia dello stack dicom sx / dx.
**Svuotato** non deve essere mai svuotato.

**Riempiti dalla routine di detection (libmigdet_2d.dll):**

**mig_lst_t det_r**
**mig_lst_t det_l** liste di punti di interesse (coordinate x,y,z e raggio) di eventuali noduli per il polmone dx e sx. Ancora sparse, non organizzate in strutture 3D.

**Riempito** dalla dll libmigdet_2d.
**Serve** alla dll di fpr1 (libmig_fpr1.dll).
**Svuotato** in parte da libmig_fpr1.dll, in parte da mig_cad.cpp.

**mig_cleanup_f det_cleanup** funzione da usare per ripulire i dati allocati dall routine di detection (libmigdet_2d.dll).

**Riempito** dalla dll libmigdet_2d.
**Serve** alla pulizia della lista dei punti d'interesse.
**Svuotato** non deve essere mai svuotato.

**Riempiti dalla routine di fpr1 / fpr2 (libmig_fpr1_.dll, libmig_fpr2.dll):**

**mig_lst_t results** risultati finali del cad.

**Riempito** dalla dll libmig_fpr1.dll , libmig_fpr2.dll.
**Serve** alla scrittura dei tag (risultati) finali.
**Svuotato** alla fine del processing da mig_cad.cpp

} mig_cad_data_t;

**NOTA**: tutte le liste di tipo **mig_lst_t** usate nella struttura **mig_cad_data_t** sono composte da nodi contenenti strutture di tipo **mig_im_region_t**.

## Struttura delle varie DLL.

Tutte le dll di processing contengono quattro funzioni:

**MIG_INIT_F_NAME** **"mig_init"**
**MIG_RUN_F_NAME** **"mig_run"**
**MIG_CLEANUP_F_NAME** **"mig_cleanup"**
**MIG_INFO_F_NAME** **"mig_info"**

**mig_init** deve essere chiamata almeno una volta prima dell'utilizzo della dll. Serve a registrare all'interno della dll l'indirizzo dei dati del cad (mig_cad_data_t) e l'indirizzo dei parametri del cad (mig_dic_t).

**mig_run** viene chiamata per ogni paziente nuovo e fa il processing dei dati di quel paziente.

**mig_cleanup** viene chiamata fare pulizia dei dati allocati dalla dll.

**mig_info** viene chiamata per avere informazioni sulla dll (versione, data di compilazione, etc.). Non utilizzata per adesso.

## Librerie precompilate (esterne).

Le libreria esterne necessarie al funzionamento del cad sono:

1. Librerie statiche dcmtk.
2. Librerie statiche sqlite3.
3. Librerie statiche libtiff.
4. Librerie statiche e dinamiche log4cplus.
5. Librerie statiche e dinamiche pthreadVC2 (per windows).
6. Librerie statiche zlib.

# Percorso dati INPUT OUTPUT

## *Caricamento DICOM*

Al momento deve essere presente la directory del paziente e un'entry per il paziente nel database.
Il campo process_status deve essere posto a 1 perché il paziente venga processato.
Viene riempito lo stack (non separato left-right).

```
Mig16u *stack
mig_size_t stack_s
```

## *Segmentazione*

Vengono creati gli stack separati e segmentati, che verranno utilizzati dalle parti successive.

```
Mig16u *stack_l(r)
mig_size_t stack_l(r)_s

mig_roi_t bb[2]    bounding boxes per il polmone dx [0] e sx [1] segmentati
```

## *Detection*

```
Usa gli stack e bb.
Riempie
mig_lst_t det_r
mig_lst_t det_l
```

con le strutture individuate dalla detection

vedi paragrafo:Il tipo che contiene le informazioni sulle ROIs

## *FPR1, FPR2*

```
Input e Output:
mig_lst_t det_r
mig_lst_t det_l
```

Nel caso di FPR1, queste liste contengono i crop 2D non raggruppati. In FPR1 vengono raggruppati in strutture 3d e vengono eliminati quelli che non soddisfano i criteri di FPR1.
All'uscita da FPR1 abbiamo det_r e det_l popolati con le strutture 3d sopravvissute. Queste diventano l'input di FPR2.
All'uscita da FPR2 det_r e det_l vengono unite in _CadData.results che contiene i risultati finali.
vedi paragrafo:Il tipo che contiene le informazioni sulle ROIs

## *Il tipo che contiene le informazioni sulle ROIs*

Il tipo utilizzato per contenere i dati della detection e delle varie fasi FPR è

```
typedef struct _mig_im_region_t
{
        float       centroid[3];    /* x , y , z centroid */
        float       radius;         /* used for circular and spherical regions :
in pixels */
        int         size;           /* size in voxels */
        mig_lst_t   objs;           /* mig_im_region_t 2D objects making up 3D
object */

} mig_im_region_t;
```

in particolare objs contiene una lista di mig_im_region_t che indica tutta la struttura 3d (uscita di FPR1 e in-out di FPR2).

# File di configurazione

Il file .ini da passare come argomento alla riga di comando del cad contiene tutte le opzioni di configurazione del software. È possibile selezionare la dll da utilizzare per ogni parte del CAD (dicom, segmentation, detection, FPR-1 e FPR-2) semplicemente inserendo il suo percorso.

```
[log]                                          ; logging system parameters
logini = "D:\LUNG_C\etc\cadlog.ini"            ; logging system configuration file

[general]                                      ; general setup parameters
dir_out   = "D:\data\results\"                  ; where to store results
db_file = "D:\data\orsola.db"                   ; database file
queue_len = 1                                  ; how many unprocessed entries to
read from database file
retry_read = 10                                ; polling interval in seconds for
unprocessed entries
retry_write = 10                               ; polling interval in seconds for
signaling an entry as processed

[io]                                           ; dicom io parameters
dll = "D:\LUNG_C\build\libmigdicom_d.dll"       ; dicom io dll to use
wc = -600                                      ; dicom loading window center
ww = 1500                                      ; dicom loading window width
resample = 0                                   ; shall we resample input stack
target_z_resolution = 0.625                    ; if resample is 1 use this as
target z resolution
dump = 1                                       ; shall we dump original / resampled
stack to a mat file

[segmentation]                                 ; segmentation parameters
perform_segmentation = 1                       ; shall we perform segmentation
dll = "D:\LUNG_C\build\libmigseg_d.dll"         ; segmentation dll to use

[segmentation/threshold]                       ; segmentation thresholding
parameters
g0 = -1000                                     ; start threshold - Hounsfield Units
g1 = 0                                         ; end threshold - Hounsfield Units
g2 = -500                                      ; fallback threshold - Hounsfield
Units

[segmentation/lung_separate]                   ; left - right lung separation
parameters
sep_min_area       = 0
sep_accum_thr_ini  = 3
sep_accum_thr_size_max  = 0.60

[segmentation/debug]                           ; segemntation debugging
dump = 63                                      ; shall we dump segmentation steps'
images
dir_dump = "D:\data\segmentation\"  ; directory where to dump segementation
steps' images

[detection]                                    ; detection parameters
perform_detection = 1                          ; shall we perform detection
dll = "D:\LUNG_C\build\libmigdet_2d_d.dll"      ; detection dll to use

[detection/radial]                              ; fast radial parameters
radii = { 1.0 , 3.0 , 5.0 , 7.0 , 9.0 , 11.0 , 13.0 }         ; fast radial list
of radii ( in pixels )
threshold = 1.00                                ; fast radial response threshold
```

```
( in percentage of the maximum value )

[detection/sspace]                                  ; scale space parameters
spacing = 1                                         ; how to calculate sigmas'
spacing : 0 geometric , 1 arithmetic progression
increment = 1.0                                     ; if spacing is 1 this gives the
step increment
min_nod_diam = 3.0                                  ; minimum nodule diameters we
are looking for ( in mm )
max_nod_diam = 20.0                                 ; maximum nodule diameters we
are looking for ( in mm )
threshold = 0.0                                     ; threshold for local maxima
detection

[detection/debug]                                   ; detection debugging
dump = 1                                            ; shall we dump detection steps'
images
dir_dump = "D:\data\detection\"         ; directory where to dump detection
steps' images

[fpr1]
perform_fpr1 = 1                                    ; shall we perform fpr1
dll = "D:\LUNG_C\build\libmigfpr_1_d.dll"           ; fpr1 dll to use

delta_tolerance = 7.75                              ; fpr1 x/y distance tolerance in
pixels
max_scan_distance = 3.0                             ; fpr1 z distance in slices

max_obj_length = 13                                 ; maximum object length along z
axis

max_angle_l2  = 72.5                                ; allowed centroid angle along z
for object of z length 2
max_angle_l3  = 66.2                                ; allowed centroid angle along z
for object of z length 3
max_angle_l4  = 66.2                                ; allowed centroid angle along z
for object of z length 4
max_angle_l5  = 66.2                                ; allowed centroid angle along z
for object of z length 5
max_angle_l6  = 52.8                                ; allowed centroid angle along z
for object of z length 6
max_angle_l7  = 38.2                                ; allowed centroid angle along z
for object of z length 7
max_angle_l8  = 38.2                                ; allowed centroid angle along z
for object of z length 8
max_angle_l9  = 38.2                                ; allowed centroid angle along z
for object of z length 9
max_angle_l10 = 16.0                                ; allowed centroid angle along z
for object of z length 10

max_obj_volume_low_l2  = 0.0                        ; allowed minium volume for
objects of length 2
max_obj_volume_low_l3  = 0.0                        ; allowed minium volume for
objects of length 3
max_obj_volume_low_l4  = 0.0                        ; allowed minium volume for
objects of length 4
max_obj_volume_low_l5  = 8.45                       ; allowed minium volume for
objects of length 5
max_obj_volume_low_l6  = 11.5                       ; allowed minium volume for
objects of length 6
max_obj_volume_low_l7  = 40.0                       ; allowed minium volume for
```

```
objects of length 7
max_obj_volume_low_l8  = 121.0                  ; allowed minium volume for
objects of length 8
max_obj_volume_low_l9  = 121.0                  ; allowed minium volume for
objects of length 9
max_obj_volume_low_l10 = 219.0                  ; allowed minium volume for
objects of length 10

max_obj_volume_high_l2  = 131.0                 ; allowed maximum volume for
objects of length 2
max_obj_volume_high_l3  = 131.0                 ; allowed maximum volume for
objects of length 3
max_obj_volume_high_l4  = 131.0                 ; allowed maximum volume for
objects of length 4
max_obj_volume_high_l5  = 131.0                 ; allowed maximum volume for
objects of length 5
max_obj_volume_high_l6  = 191.0                 ; allowed maximum volume for
objects of length 6
max_obj_volume_high_l7  = 218.0                 ; allowed maximum volume for
objects of length 7
max_obj_volume_high_l8  = 218.0                 ; allowed maximum volume for
objects of length 8
max_obj_volume_high_l9  = 330.0                 ; allowed maximum volume for
objects of length 9
max_obj_volume_high_l10 = 330.0                 ; allowed maximum volume for
objects of length 10

[fpr2]
perform_fpr2 = 0                                ; shall we perform fpr2
dll = "D:\LUNG_C\build\libmigfpr_2_d.dll"           ; fpr2 dll to use

svm_model_file = svm.model                      ; libsvm format model file
svm_norm_file  = svm.nrom                       ; feature normalization
parameters

rotations = 24                                  ; number of 2d rotations to
perform on each 3d object view
resized_len = 19                                ; resize crops to this side
length for classification

min_rot_pos_l2  = 13                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l3  = 13                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l4  = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l5  = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l6  = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l7  = 13                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l8  = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
```

```
min_rot_pos_l9  = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l10 = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l11 = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l12 = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l13 = 14                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l14 = 12                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght
min_rot_pos_l15 = 11                            ; min number of rotated views
that must have a positive classification label to consider view as positive for
this 3d lenght

min_pos_views_l2  = 2                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l3  = 1                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l4  = 1                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l5  = 1                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l6  = 2                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l7  = 4                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l8  = 5                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l9  = 5                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l10 = 5                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l11 = 5                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l12 = 6                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l13 = 7                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
min_pos_views_l14 = 6                           ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
```

```
min_pos_views_l15 = 5                              ; minimum number of 2d views
composing 3d object that must have had a positive classification to consider 3d
object as positive for this length
```

# Differenze fra livello concettuale e sviluppo.

- Il raggruppamento delle ROIs in strutture tridimensionali avviene all'interno della dll FPR-1;
- SVR e Ranklet 3D non sono ancora integrate;
- ....