Verslag Meeting V

Do 17/04/2014 (uitgesteld van 10/04/2014)

Aanwezig

- Dr.ir. Pieter Slagmolen
- Ir. David Robben
- Dr. Yoni De Witte
- Ir. Sven Van Hove
- Ir. Kim Nuyts

Agenda

Vooruitgang sinds vorige meeting

Longsegmentatie

- Doel: ruwe aflijning om rekentijd in te korten
    - Vraag: hoeveel pixels gooi je er dan uit? REKEN UIT! Schatting: 50%
- Vorige algoritme te complex
- Test constante threshold: OK
- Methode:
    - Threshold (1600) om de chest wall van de rest te scheiden
    - Morfologische opening om kleine blobs te verwijderen
    - Morfologische reconstructie om longen op te vullen
    - Morfologische erosie om het te grote gebied wat kleiner te maken
- ➔ FixedThresholdTest.py
- TODO: zet threshold van 2D naar 3D implementatie
    - Probleem: erosie wil niet meer werken in 3D

Aflijning nodules a.h.v. vertices

- Cirkel per slice: OK
- Veelhoek per slice: OK
- Volume
    - Ellipsoïde in pixel coordinates
    - Omzetten naar wereldcoordinaten -> bol
        - Nog altijd Rz << Rx,Ry ???
        - Efficiëntieproblemen
    - Toch beter formule ellipsoïde gebruiken, of cirkel per slice

Suggesties Pieter:

- gebruik niet midden ([Xmax – Xmin]/2) maar bereken zwaartepunt → euclidische afstand
- geen overlap, noch spaties tussen slices

- hebben we de aflijning van de nodules als bollen nodig?
    - Eventuele meerwaarde: bereken de gradiënt in intensiteit naar buitenkant van bol toe als feature voor malignancy
    - Nee, betere oplossing: neem polygoon per nodule en bereken zwaartepunt (niet midden) en doe dan erosie van de polygoon om buitenste rand te elimineren
- Werken met enkel pixels in 'midden' zou geen probleem met 'holle nodules' mogen geven zolang we niet enkel intensiteit van individuele pixels als feature gebruiken

Implementatie features

- Features die geen a priori nodule segmentatie nodig hebben
- Volume Features (3D)
    - Edges (Sobel) -> window
- Slice Features (2D)
    - Entropy -> zwaar
    - Blobs (max LoG)
        - TODO: 3D (voxelgrootte!) -> beter op kleiner window (om geheugen te sparen)
        - Gebruik enkel scipy library: scipy.ndimage.convolve (wel 3D)
- Pixel Features
    - Intensity
    - Location (absolute, relative)
    - Frobenius Norm (distance to center)
    - Neighbours (L-R, T-B, …)
- Window Features (n = 3:32:2)
    - Nodule -> window size 50 -> traag
      MAAR wel gebruiken om ook grote nodules te segmenteren! Eventueel pas later in cascade? Eventueel meer stappen ertussen laten?
    - Intensity (mean, variance, skewness, kurtosis, max, min, …)
    - Left/Right/Top/Bottom/Front/Back row -> mean, grad, …
        - Implementeren met convolutiefilters?
- ➔ featureselection.py

- TODO: entropy, haar, …
    - Haar features
        - SimpleCV werkt niet goed in Python (David) -> zelf programmeren (numpy)
        - Pas laat in cascade gaan gebruiken
    - Entropy: pas laat in cascade gaan gebruiken (wel snel berekend op hele image)
- Cascading? Gebruik predict_proba (geeft rijen = samples en kolommen = klassen met per klasse een probabiliteit)
    - Niveau 1: Gebruik bv 1 simpele feature (intensiteit) en schrap daarmee deel van de trainingsdataset (alle pixels die 99% zeker geen nodule zijn eruit halen)
    - Niveau 2: gebruik iets gesofisticeerdere features en elimineer weer deel van de dataset
    - Niveau 3: idem
    - …

Classificatie

- Eerst: trainen met enkel nodule pixels binnen cirkel r/3: OK (triviaal), maar traag
- Vervolg: non-nodule pixels samplen (welke?)
    - TRAININGSDATASET (Pieter): gebruik helft pos, helft neg + sample random in 3D

⇨ Genereer **een probabilistisch beeld** na elke stap in de cascade en bekijk of het een goede stap is of niet (= meer 1 en minder 0 voxels? En behouden we alle nodules of moeten we extra features toevoegen om alle nodules te behouden?)

- Vgl ook steeds met Mevislab (.jpeg, .dicom, .tif, .nifty… pydicom)

⇨ THRESHOLDING mogelijk niet meer nodig indien de eerste classificatiestap er al alle achtergrond pixels uitsmijt

Planning komende week

- Lichaamsegmentatie uitbreiden naar alle slices (slice per slice of 3D?)
- Basisclassificatie uittesten met zowel nodule als non-nodule pixels
- Cascaded classififer implementeren
- Meer features implementeren (haar, entropy, ...) Suggesties?
    - Beter eerst huidige features analyseren

Vragen & Opmerkingen

- Welke pixels uitkiezen voor
    - nodule training
    - non-nodule training -> willekeurig! Maar niet te dicht in buurt van nodules
- We hebben nu meer dan 500 features (met windowvariaties inbegrepen) die van elke voxel in de scan moeten bepaald worden. Tot nu toe hebben we het proces versneld door de data te downsampelen, maar zelfs dan duurt het nog redelijk lang: een paar minuten voor 1 nodule. Daarom hadden we onze maximum windowgroottes ook al beperkt (van 50 tot 30). Het algoritme op alle pixels met alle windowgroottes te laten draaien zal veel tijd vragen.
    - Ligt deze rekentijd in de lijn van verwachtingen? Of doen we iets fout waardoor onze rekentijd onredelijk lang wordt?
    - Voor het trainen van de classifier zijn er volgens ons twee mogelijkheden. Ofwel sampelen we de scan en nemen we om de zoveel pixel er een uit die op 0 (geen nodule) of op 1 (nodule) kan gezet worden. Ofwel gaan we expliciet op zoek naar de positieve pixels en berekenen we de negatieve apart. (hier ontstond ook wat verwarring: voorlopig werken we met een cirkel per slice)
        - ➔ windowgroottes minder laten varieren in het begin
- Bij het resizen van onze dataset (omwille van bovenstaande efficiëntieredenen) zien we ook dat onze nodulemasks helemaal vervormd worden.
    - ➔ Door subsampelen dataset: mogelijk voor developing, in def versie zeker niet!! En let op, want zo verlies je mogelijk info over de kleine nodules
    - ➔ Neem liever ROI rond nodule (beetje long, beetje bot, beetje spier …) om sneller development te kunnen doen
- De Haar Features: We vinden een library van SimpleCV die met deze haar features werkt. Python doet echter moeilijk als we deze willen inladen. Is er een beter pakket hiervoor?

- Entropy feature: We hebben een feature die de entropy van de afbeelding bepaald. Hiervoor moeten we echter een .view('uint8') commando gebruiken. Hier ondervinden we ook problemen mee. (from skimage.filter.rank import entropy)
    - ➔ Gebruik NIET .view, gebruik WEL .astype('uint8')
- Cascaded feature selection: Hoe pakken we dit concreet aan?
- We hebben twee methodes gevonden om output te genereren met ExtraTrees. Daarvan hebben we deze gekozen omdat we dachten dat die probabiliteiten als uitkomst zou geven, maar er is geen verschil in resultaat merkbaar:
    - result = clf.predict_proba(X[0,:])
- Voorlopig sampelen we door de slices met nodules in om een trainingsset te genereren: pixels worden op 0 (geen nodule) of 1 (nodule) gezet. Sampelen we best ook door alle andere slices om ook daar de robuustheid tegen FP te verhogen? Dit gaat dan wel ten koste van de rekentijd.
    - We dachten de rekentijd te beperken door enkel door die slices te sampelen waar nodules inzitten. Dit zijn de belangrijkste voor de radioloog om het aantal FP's in te reduceren (in onze ogen). Een radioloog die een FP ter hoogte van de lever krijgt, ziet dat zelf ook meteen en kan die dus negeren. Maar een FP op een plaats waar je nodules redelijkerwijs kan verwachten, willen we wel zeker vermijden.

Volgende meeting

Donderdag 24/04/2014 @ MIRC, 17u00