Supplemental data for the paper:

Methods for motion correction evaluation using FDG human brain scans on a high resolution PET scanner

by Sune H. Keller, Merence Sibomana, Oline V. Olesen, Claus Svarer, Søren Holm, Flemming L. Andersen, and Liselotte Højgaard

This material consists of three parts giving further details on 1) the motion correction method EMT, and the two motion correction quality control methods 2) mutual information and 3) cross correlation.

External motion tracking - EMT

Motion tracking

An optical 3D motion tracking system, Polaris Vicra (NDI), was used to register head movements during the scans. The system uses a rigid tracking tool with 3 markers, and the system measures the orientation and position of the tool The tracking tool was fixed to the forehead along the hair line where skin movements are limited. The fixation method uses a standard band-aid with velcro tape from Apodan Nordic.

The transformation between the coordinate system of the HRRT *image frame* and the coordinate system of the Polaris *tracker coordinate system* has to be established to enable MC in image space using Polaris tracking data. For that we have used a customized reference tool fixed inside the tunnel close to the patient tool and a one time calibration with 10 paired high statistics transmission (TX) scans and Polaris trackings of the patient tool fitted with 4 markers. The 3D rigid transformation between the two coordinate systems was found by identifying the same set of points (the center of the markers of the tool) and then using the closed-form loop solution to estimate the absolute transformation.

Automated frame division

The tracking data were divided into subframes of a given time interval *dt* and tested for inter-frame motion (Fig. 1). If a subframe had significant motion (larger than *m* millimeter) it was treated as a preserved subframe. After motion correction the subframes were joined into the original frames.

The analysis of the tracking was as follows: The tool was chosen as the test point, thus the recorded translations (T_x, T_y, T_z) were evaluated representing the motion of the brain. This is of course not the true motion of the complete brain. However, the point in the forehead is highly sensitive to motions, since two primary head motions during scans are a) rotation around the contact point at the back of the head, and b) translation of the head in the axial axis direction. The relative motion of the tool to a reference position was estimated and low pass filtered (cut of at 0.5 Hz). Then the filtered motion was divided into subframes of $dt = 10$ s and tested for intraframe motion. An *F*-test (two-sided) assuming two independent random samples was performed.

A second criterion had to be fulfilled in order to preserve a subframe for intraframe motion. The difference between the mean positions of the subframe itself and the prior frame had to be $>m$. For

this study *m* was set to 1 mm.

The preserved subframes were found with respect to two reference positions: a) We decide if the μ-map should be aligned to the EM frames prior to reconstruction. Thus the tool position during the first 60s of the TX scan is used as a reference position. b) We decide if the reconstructed EM frames should be aligned to the reference EM frame suggested by our algorithm, and thus this EM reference frame is used as a second reference position when estimating the subframing.

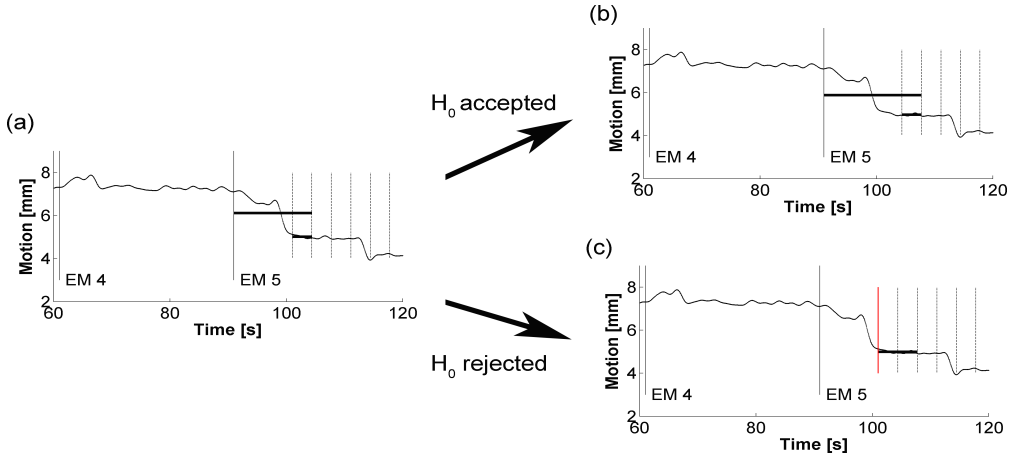

Fig. 1. Principle of the test for intraframe motion. EM dynamic frames 4 and 5 showing the relative tracked motion of the tool during the test for intraframe motions within frame 5. Vertical black lines: EM frame division. Short vertical black lines: test of subframe division. Red line in (c): preserved subframing due to significant intraframe motion.

Motion corrected reconstruction

The principle of the performed frame re-positioning motion correction was: 1) The EM list-mode data were histogrammed into frames based on the predefined EM framing and the subframes created due to intraframe motion. 2) The μ-map was aligned with each EM frame if the tracked motion during a given EM frame and the TX scan was significant different and larger than 1 or 2 mm based on the same two-sided *F*-test as for the detection of intraframe motion. 3) The EM frames were reconstructed using 3D-OSEM PSF (16 subsets, 10 iterations). 4) The EM frames were aligned to a reference frame if the difference between the actual EM frame and the reference frame was higher than 1 mm. 5) The subframes with intraframe motion were combined to give the original EM framing.

Frame of reference

The method that analyzes the tracked head position also suggests a frame of reference found by a score-function with four subfunctions. The function return a score for each EM frame depending on the variables of the subfunctions: 1) duration of the frame $S_L(t)$, 2) root mean square error (RMSE) of the tracking from a mean position $S_E(rms)$, 3) time from the injection normalized to the EM scan duration $S_I(t^*)$, and 4) the distance to a mean position of the complete EM scan $S_P(p)$. Fig. 2 shows the four subfunctions and the score-function was defined as

$$S(frame) = S_L(t)[S_E(rms)\cdot w_E + S_I(t^*)\cdot w_I\, S_P(p)\cdot w_P]$$

where w_E, w_I, and w_P were weights that could be adjusted as wanted. The returned suggested frame of reference was the frame with the highest score and no subframes if possible.

Additional detail on EMT can be found in the paper:

Olesen OV, Keller SH, Sibomana M, Larsen R, Roed B, Højgaard L. Automatic thresholding for frame-repositioning using external tracking. *IEEE Nucl Sci Symp Conf Rec*. 2010:2669-2675.

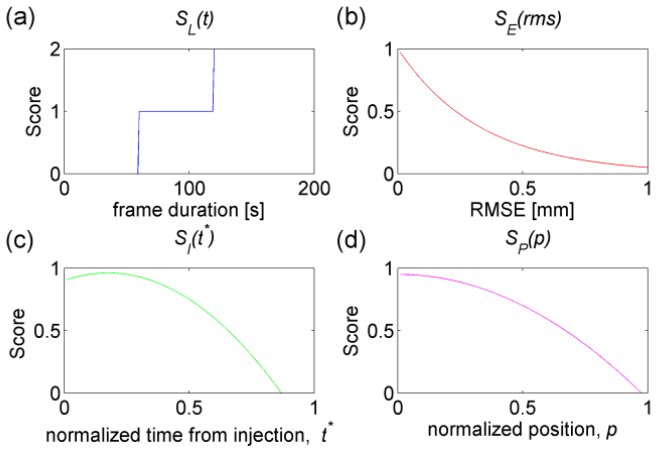

Fig. 2. Subfunctions of the score-function S(frame). The four plots show the score as a function of: (a) frame duration, (b) RMSE of the tracking, (c) time from injection, and (d) distance to the reference position.

Mutual information – MI

To give a detailed insight into our use of mutual information, we give our Matlab code used here:

```
function MI = mutualInfoCut(Im1, Im2, bins, cut)
% Function to compute mutual information of two 3D image volumes.
%
% Syntax: MI = mutualInfo(Im1, Im2, bins, cut)
% MI is the computed mutual information.
% Im1 and Im2 are the two images to be compared.
% bins are the number of bins in the MI histograms. It is a fairly sensitive
% parameter and should be set carefully. 256 is recommended.
% cut is the number of lower bins to cut to mask out image background.
%
% This function is not a registration tool. It is a QC tool that computes
% the MI of a given image pair. Comparing the MI of several images pair
% will give a measure of which pair that has the highest MI (and thus are
% best registered to each other).
%
% Required preprocessing: Load image volumes into Matlab and scale to same
% size.
%
% Written by Sune Høgild Keller and Merence Sibomana, HRRT Users Community,
% Rigshospitalet, 02-09-2010.

% get size of images
[m,n,p] = size(Im1);
```

```
% Rescale grey value range of images to the bin range [0, bins-1].
% 1. set min to 0
Min1= min(Im1(:));
Im1 = Im1 - Min1;
% Scale max to 'bins'-1 and move values from float[0, bins-1] to int[1, bins]
Max1= max(Im1(:));
Max1 = (bins-1)/Max1;
Im1 = round(Im1*Max1)+1;

% Im 2, same procedure
Min2= min(Im2(:));
Im2 = Im2 - Min2;
Max2= max(Im2(:));
Max2 = (bins-1)/Max2;
Im2 = round(Im2*Max2)+1;

% Initialize histograms as ones to avoid zero bins, and okay as it is the
% distribution that is interesting, not the absolute number value
hist2D = ones(bins);
histIm1 = ones(bins,1);
histIm2 = ones(bins,1);

% compute the actual histograms
for x = 1:m
    for y = 1:n
        for z = 1:p
            % image intensity values (rounded to int) are coordinates in the
            % histograms
            hist2D(Im1(x,y,z),Im2(x,y,z)) = hist2D(Im1(x,y,z),Im2(x,y,z)) + 1;
            histIm1(Im1(x,y,z)) = histIm1(Im1(x,y,z)) +1;
            histIm2(Im2(x,y,z)) = histIm2(Im2(x,y,z)) +1;
        end
    end
end

% Masking the histograms by cutting the 'cut' lower bins
hist2D = hist2D((cut+1):bins,(cut+1):bins);
histIm1 = histIm1((cut+1):bins);
histIm2 = histIm2((cut+1):bins);

% Normalization of the histograms to sum to 1 and thus represent the
% distributions that are the real input to the mutual information computation
hist2D =  hist2D/sum(hist2D(:));
histIm1 = histIm1/sum(histIm1(:));
histIm2 = histIm2/sum(histIm2(:));

% MI (using 3. version of Mi, which is eq. (7) from from Pluim etal. TMI 03,
% a Kullback-Leibler analog.
MI = 0;
for a = 1:(bins-cut)
    for b = 1:(bins-cut)
        MI = MI + hist2D(a,b)*log(hist2D(a,b)/(histIm1(a)*histIm2(b)));
    end
end
```

Cross correlation – XC

As for mutual information we will give the Matlab code for cross correlation. This code is a bit more advanced than the MI code as it takes a full sequence of scan frames as input.

```
function XC = crosscorrSequenceMasked3D(ImVol, refFrame, mask)
% Compute the cross correlation of each frame with the ref frame using only
% values above the threshold (inside head/brain).
%
% ImVol is the set of scan images to be processed.
% refFrame is the frame of reference to which each frame is compared to get
% the XC.
% mask is a binary mask excluding background voxels from the computations
% of the XC.
%
% Preprocessing: Load image volume into Matlab and compute the mask by simple
% thresholding. We suggest to make the mask on the ref frame image smoothed
% with a 6mm Gaussian.
%
% Written by Sune Keller and Merence Sibomana, HRRT Users Community,
% Rigshospitalet, 31 Aug 2010

% The number of voxels in the mask for normalization.
total_mask_vol = sum(mask(:));

% Reference frame preprocessing:
% Use values inside the mask only.
Xr = ImVol(:,:,:,refFrame).*mask;
% Compute the mean inside the mask.
br = sum(Xr(:));
mr = br/total_mask_vol;

% Loop over the frames to preprocess and compute the XC for each of them.
for k = 1:size(ImVol,4)
    %Frame preprocessing:
    % Use values inside the mask only.
    X = ImVol(:,:,:,k).*mask;
    % Compute the mean inside the mask.
    b = sum(X(:));
    m = b/total_mask_vol;

    % A vector of the indexes of all voxels included in the mask.
    idx = find(mask == 1);

    % 3D cross-correlation of voxels in the mask only.
    X1 = X-m;
    Xr1 = Xr-mr;
    XC(k) =
sum(X1(idx).*Xr1(idx))/sqrt(sum(X1(idx).*X1(idx))*sum(Xr1(idx).*Xr1(idx)));
end
```