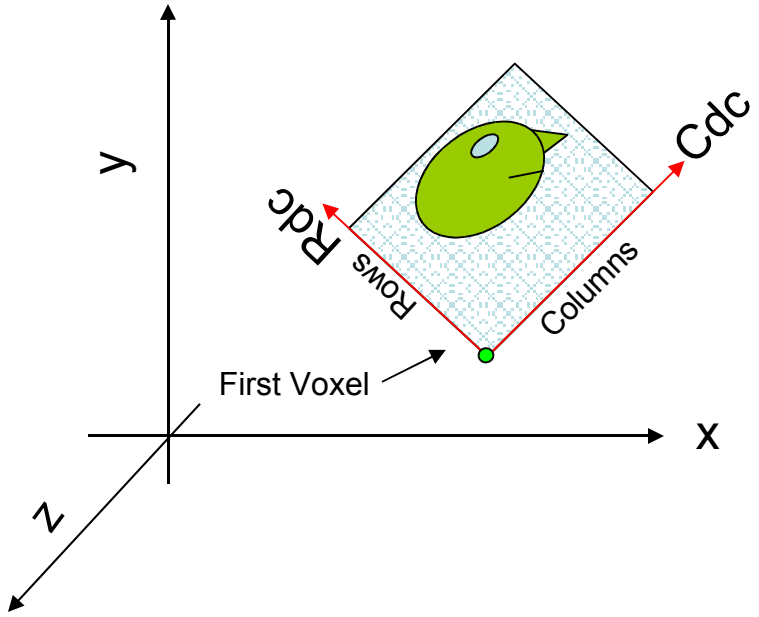
Theory of Affine Spatial Transforms:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = [Mdc*D\ P0] * \begin{bmatrix} C \\ R \\ S \\ 1 \end{bmatrix}$$

One needed here because affine.



CRS = Column-Row-Slice
V = [Mdc*D P0] =
V: Vox2XYZ matrix (cf Vox2RAS)

Matrix of Direction Cosines (3x3)
Mdc = [Cdc Rdc Sdc]
Direction Cosine Vectors:
Cdc = [Cx Cy Cz]'
Rdc = [Rx Ry Rz]'
Sdc = [Sx Sy Sz]'

Field of View defined by:
Voxel Sizes: dC, dR, dS
Direction Cosines: Cdc Rdc Sdc
First Voxel: X0 Y0 Z0

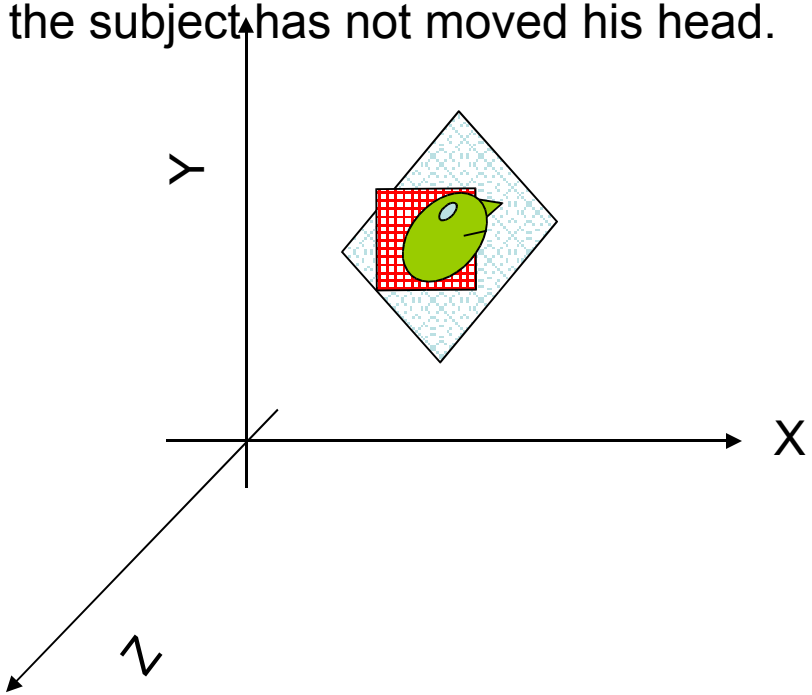
Matrix of Voxel Sizes
D = diag([dC dR dS])
dC – column size
dR = row size
dS = slice thickness

Note Arbitrariness: eg, one can change the first voxel and some of the other parameters to create a new correct vox2XYZ matrix (Tower of Babel).

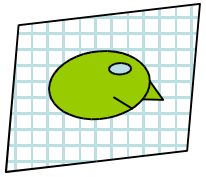
First Voxel – XYZ at CRS=0
P0 = [X0 Y0 Z0]';

Theory of Affine Spatial Transforms: Two Fields of View within One Coordinate System

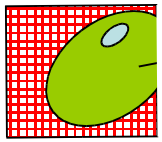
To map from one FoV to the other, need to know the correspondence between the col, row, slice in one FoV and the other, ie, given the CRS in Volume 1, what is the CRS in Volume 2? We call this a “vox2vox” transform. An example of this problem
Is when a functional and structural have been acquired in the same session and the subject has not moved his head.



Volume 1



Volume 2



XYZ=V1*CRS1

XYZ=V2*CRS2

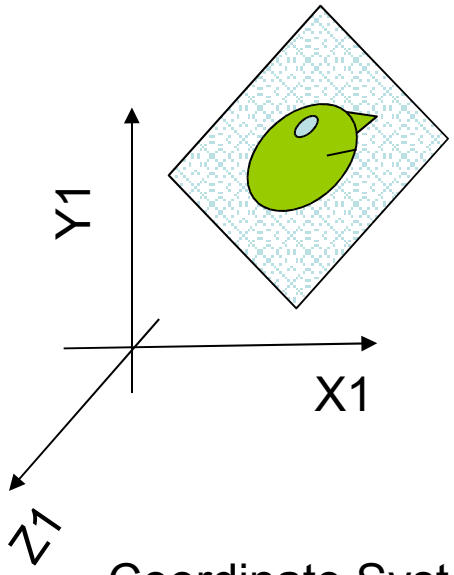
XYZ is the same for both, so

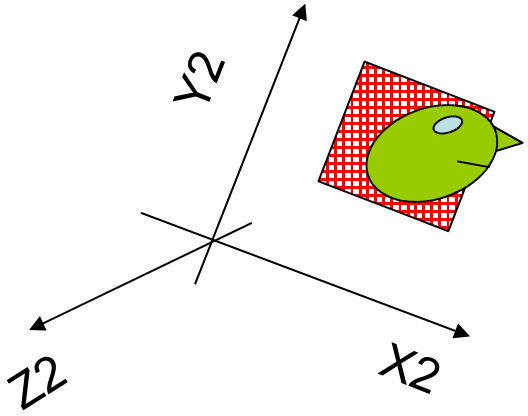
XYZ=V1*CRS1=V2*CRS2

CRS2 = [inv(V2)*V1] * CRS1
CRS1 = [inv(V1)*V2] * CRS2

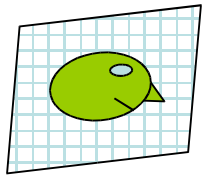
Note that [inv(V2)*V1] and [inv(V1)*V2] are “Vox2Vox” matrices.

Theory of Affine Spatial Transforms: Two Fields of View, Two Coordinate Systems



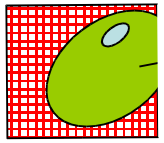


Volume 1



XYZ1=V1*CRS1

Volume 2



XYZ2=V2*CRS2

Coordinate Systems are not
the same but are linearly related:
XYZ2 = M*XYZ1 (M is registration)
XYZ1 = V1*CRS1 = inv(M)*XYZ2

V1*CRS1 = inv(M)*V2*CRS2
CRS1 = [inv(V1)*inv(M)*V2] * CRS2
CRS2 = [inv(V2)*M*V1] * CRS2

Note that [inv(V1)*inv(M)*V2] and [inv(V2)*M*V1] are Vox2Vox matrices. These are NOT arbitrary! BUT, V1, V2, and M can change definition without changing the Vox2Vox. There is a many to one relationship between V1, V2, M and the Vox2Vox.

Where Does the Vox2XYZ matrix come from? Or, equivalently, how is the space defined? How does XYZ relate to RAS?

- Scanner Space – coordinate center at magnet isocenter, bore axis is Z, X to the left, Y to the ceiling. Direction cosines and P0 defined in DICOM file (note: this is an LPS, not RAS coordinate system (assuming subject is Head-First-Supine (HFS)).
- Native – basically the same as scanner, but RAS.
- Talairach – coordinate center at Anterior Commisure, X to the right, Y points out of the noise, Z points out of the top of the head. This is an RAS space.
- RAS – Right-Anterior-Superior (anatomical coordinates).
- Field-of-View Based – arbitrarily choose center, directions arbitrary along the index axes. No constraints in relation to RAS.
    - FSL – center at the corner of the FoV
    - FreeSurfer (tkregister) – center at N/2
- MNI, SPM – use whatever is defined in the file format (eg, QForm/SForm)

To get the Scanner Vox2XYZ (or Vox2RAS):

```
mri_info –vox2ras vol.{mgz,mgh,img,nii,nii.gz,bhdr}
```

To get the FreeSurfer/tkregister Vox2XYZ (or Vox2RAS):

```
mri_info –vox2ras-tkr vol.{mgz,mgh,img,nii,nii.gz,bhdr}
```

FreeSurfer (tkregister) Definitions (needed to interpret register.dat).

$$Cdc = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \quad Rdc = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad Sdc = \begin{bmatrix} 0 \\ +1 \\ 0 \end{bmatrix} \quad P0 = \begin{bmatrix} +Nc/2 \\ -Ns/2 \\ +Nr/2 \end{bmatrix}$$

tkregVox2XYZ (tkregVox2RAS):

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -dC & 0 & 0 & +Nc/2 \\ 0 & 0 & +dS & -Ns/2 \\ 0 & -dR & 0 & +Nr/2 \end{bmatrix}$$

Voxel Sizes: dC, dR, and dS

Field-of-View based – only depends on the identity of columns, rows, and slices, the number of voxels in each dimension, and their sizes. Unrelated to the "true" geometry but is an RAS coordinate system when the volume is "Coronally" sliced, which is the based "conformed" orientation in FreeSurfer.

To get the FreeSurfer/tkregister Vox2XYZ (or Vox2RAS):
mri_info –vox2ras-tkr vol.{mgz,mgh,img,nii,nii.gz,bhdr}

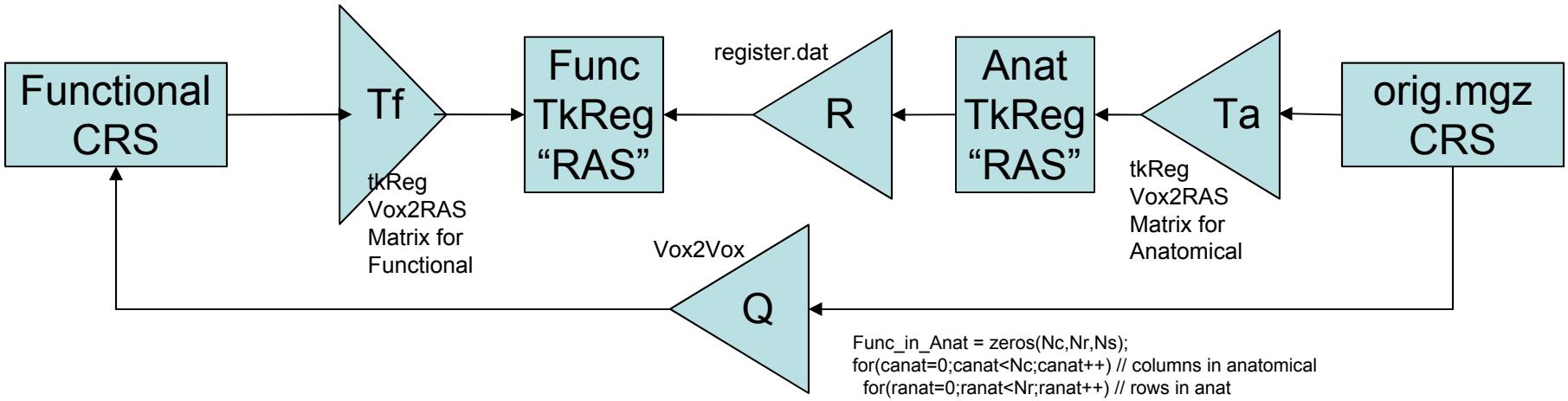
Inside the FreeSurfer C code MRI structure (mri.h):
Cdc = [x_r x_a x_s]', Rdc = [y_r y_a y_s]', Sdc = [z_r z_a z_s]'
dC = xsize, dR = ysize, dS = zsize
P0 not stored explicitly, rather the XYZ at Ni/2 is stored as c_r c_a c_s
Nc = width, Nr = height, Ns = depth

FreeSurfer (tkregister) Definitions – Mapping one volume to another with the register.dat.

Contents of register.dat:

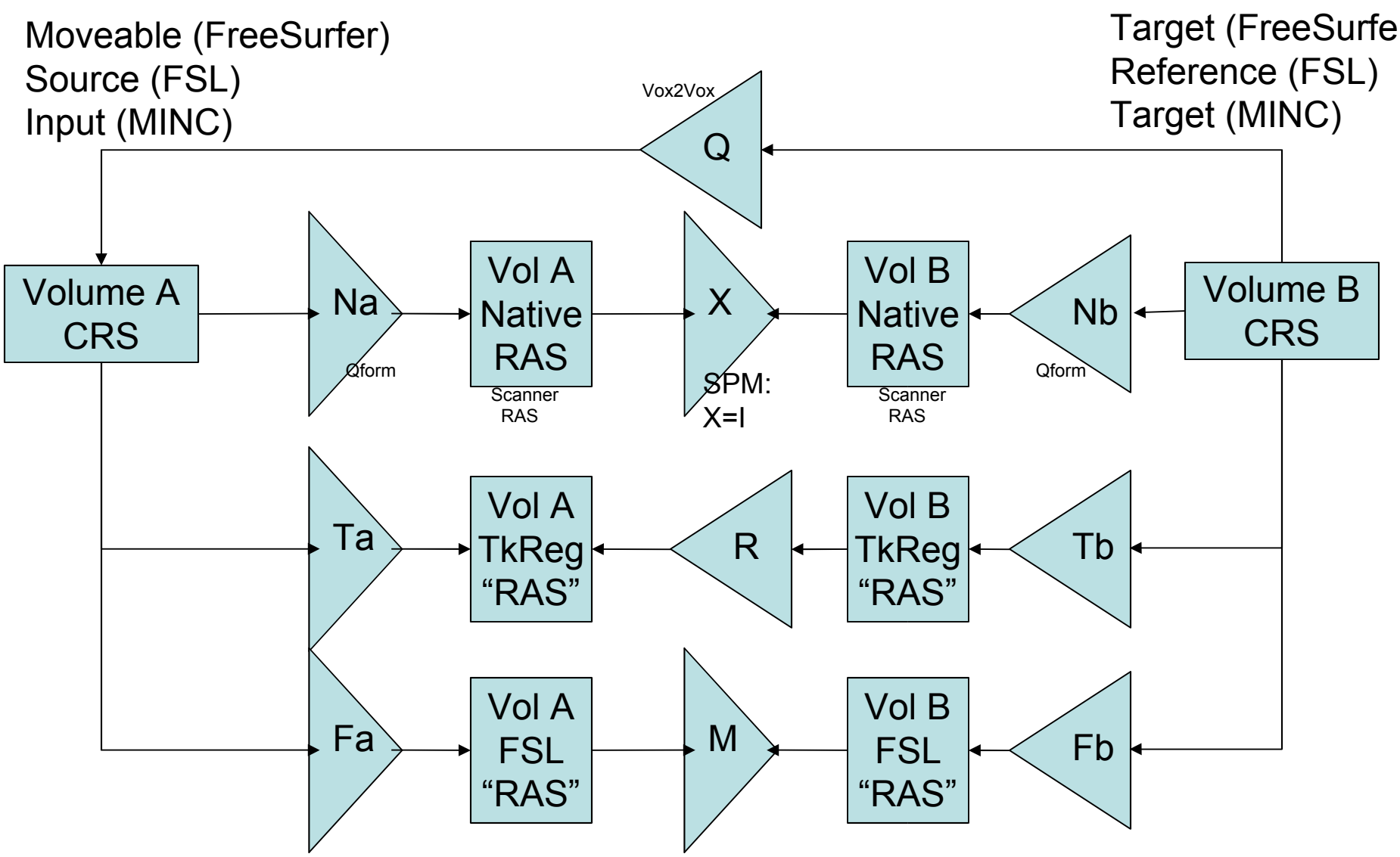
```
Subjectname
PixelSize
SliceThickness
Intensity
R11 R12 R13 R14
R21 R22 R23 R24
R31 R32 R33 R34
   0     0     0     1
round
```

SubjectName – FreeSurfer Subject ID
PixelSize – obsolete, can be ignored
SliceThickness -- obsolete, can be ignored
Intensity – only used for display in tkregister2
Rij – components of the registration matrix
round – historical accident, original tkregister performed a "floor" when computing a voxel index.



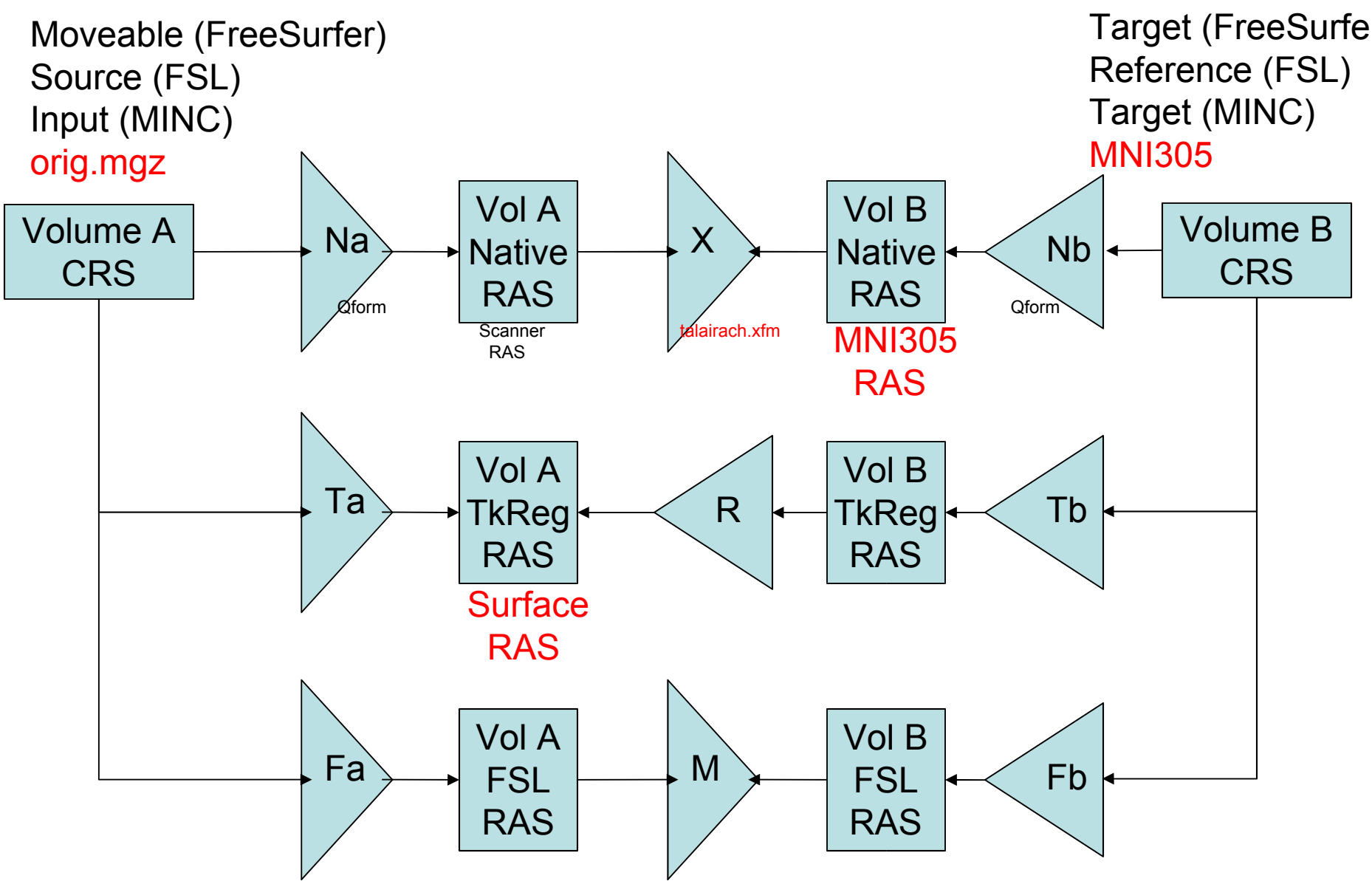
Vox2Vox Matrix: Q=inv(Tf)*R*Ta

```
Func_in_Anat = zeros(Nc,Nr,Ns);
for(canat=0;canat<Nc;canat++) // columns in anatomical
  for(ranat=0;ranat<Nr;ranat++) // rows in anat
    for(sanat=0;sanat<Ns;sanat++) // slices in anat
      anatCRS[0] = canat; anatCRS[1] = ranat;… // load into a vector
      funcCRS = Q*anatCRS; // compute indices in functional
      cfunc = round(funcCRS[0]); tfunc = round(funcCRS[1]); …
      Func_in_Anat[canat][ranat][sanat]= Func[cfunc][rfunc][sfunc];
```



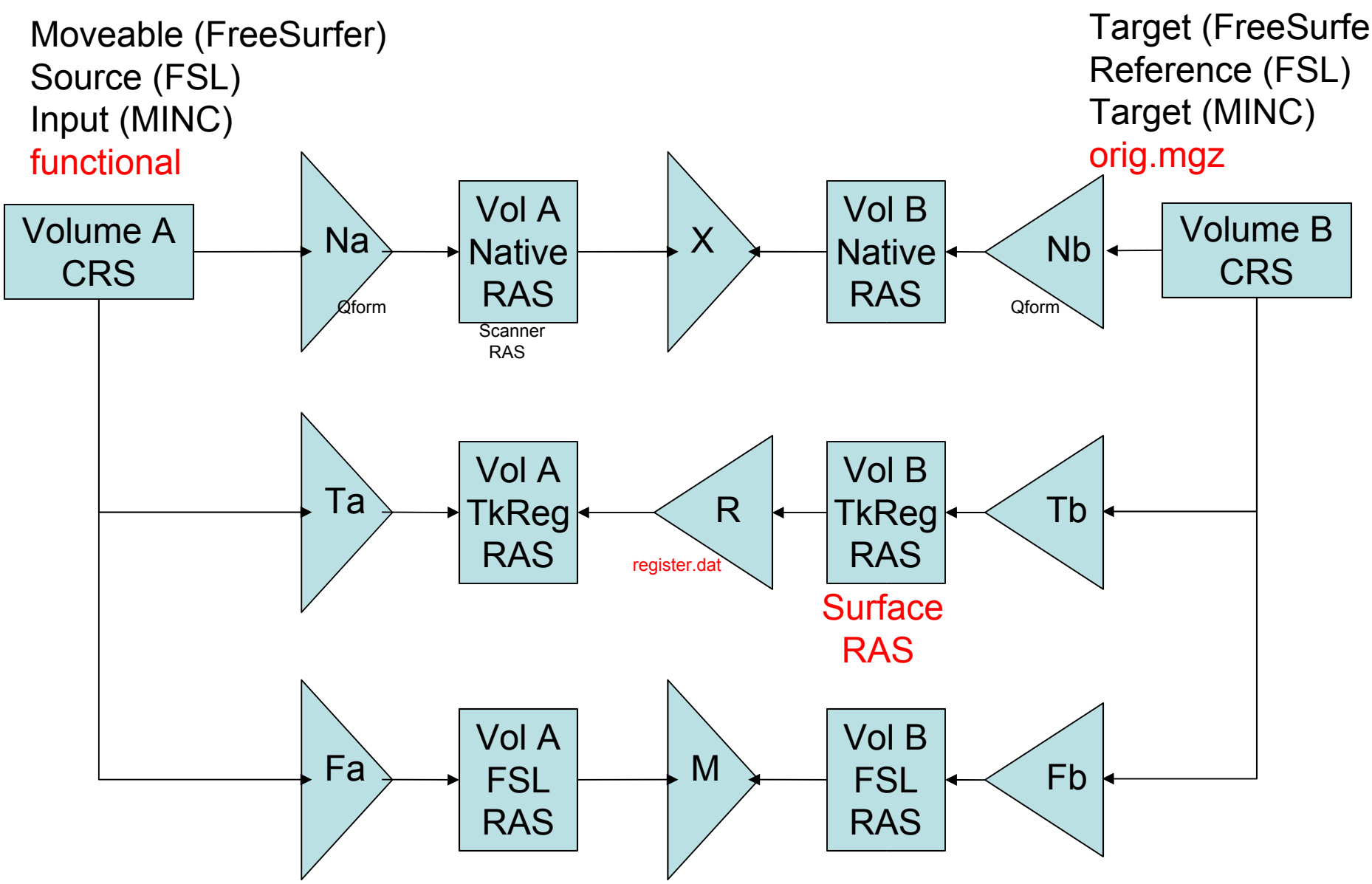
CRS = col, row, slice (voxel indices).
RAS = right, ant, sup
N? – Native or Scanner Vox2RAS
T? – TkRegister or Surface Vox2RAS
F? – FSL Vox2RAS

R = Ta*inv(Fa)*inv(M)*Fb*inv(Tb)
R = Ta*inv(Na)*inv(X)*Nb*inv(Tb)
X = Nb*inv(Tb)*inv(R)*Ta*inv(Na)
M = Fb*inv(Tb)*inv(R)*Ta*inv(Fa)



N? – Native or Scanner Vox2RAS
T? – TkRegister or Surface Vox2RAS
F? – FSL Vox2RAS

DevolveXFM():
X2 = X*Na*inv(Ta)



N? – Native or Scanner Vox2RAS
T? – TkRegister or Surface Vox2RAS
F? – FSL Vox2RAS

R = Ta*inv(Fa)*inv(M)*Fb*inv(Tb)
X = Nb*inv(Tb)*inv(R)*Ta*inv(Na)
M = Fb*inv(Tb)*inv(R)*Ta*inv(Fa)