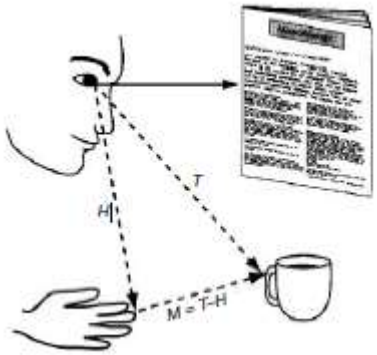
Reference Frame analysis – Tutorial

(Pieter Medendorp)

Introduction

To plan and prepare a reaching movement, information about the position of the target and information about the current position of the hand and arm must be integrated before a motor program can be formulated that brings the hand toward the target. One inherent complexity is that this information is encoded in different reference frames, which puts constraints on the computation of the difference vector between the position of the hand and the position of the target. In this tutorial we will examine modeling approaches that have been used to address this question at neural, behavioral and computational levels.



Figur1 1: from Buneo et al. Nature, 2002

This practical considers three studies that have been published in the literature on reference frame transformation:

Buneo et al. (Nature, 2002) on neural reference frames in posterior parietal cortex for reaching

Beurze et al. (J Neurophsysiology, 2006) on behavioral reference frames for reaching movements

McGuire and Sabes (Nature Neuroscience, 2009) on optimal weighting of reference frames in reach planning.

Neural reference frames

Buneo et al. (2002) approach the problem of computing the difference vector (Figure 1) by analyzing the reach-related activity of neurons in the posterior parietal cortex, while varying target position, hand position and gaze direction. Let's consider the firing rate of an idealized target position neuron:

$$f = e^{-\left(\frac{T^2}{2}\right)}$$

where T is the horizontal position of the target in eye coordinates, H is the horizontal position of the hand in eye coordinates.

Use Matlab to plot the response field (for hand and target positons) of this idealized neuron of target position, type:

```
T=[-36:1:36];      % target location relative to eye
H=[-36:1:36];      % hand location relative to eye
[t,h] = meshgrid(T/18,H/18);
f=exp(-t.^2/2);
surf(T,H,f); shading interp
title('Response Field')
xlabel('Target Location (deg)'); ylabel('Hand location (deg)');
```

```
axis([-36 36 -36 36 0 1]);
```

Take a 2D view of the response field <view(0,90)> and examine how the response tuning curve changes as function of hand and target location. Next, open a new figure to plot tuning curves as slices though each response field at initial hand locations of 0º and 36º.

```
figure, hold on
plot(T,f(1,:),'k-'); % hand location at 0 degrees
plot(T,f(37,:),'k-'); % hand location at 36 degrees
axis([-36 36 0 1]);
title('Tuning Curve')
xlabel('Target Location (deg)'); ylabel('Hand location (deg)');
```

To further characterize the response field, Buneo et al. determined the 'orientation' of the response field by computing the gradient. The gradient resultant is an indicator of the variable or variables to which the neuron is most responsive (target position, initial hand position, etc).

Open a new figure and plot the gradient the response field, as follows,

```
figure, hold on
[px,py] = gradient(f,1,1);
contour(T,H,f)          % contour lines
quiver(T,H,px,py);      % gradient field
title('gradient plot');
xlabel('Target Location (deg)'); ylabel('Hand location (deg)');
```

Zoom in to check the orientation of the gradient vectors. To account for symmetrically shaped response, we take the four quadrant arctangent of gradient field to allow for a 0º – 360º range, and plot the resultant vector.

```
phi=atan2d(px,py)*2+90;       % compute orientation of gradient vectors
phi=mean2(phi(~isnan(phi)));  % take the mean orientation
phi(phi<0)=phi(phi<0)+360     % put in range 0 to 360 degrees

figure, hold on;
quiver(0,0, sind(phi), cosd(phi)); % resultant gradient
plot(sind(0:1:360),cosd(0:1:360),'r-');     % plot circle
axis([-1 1 -1 1]); axis square; axis off;
title('gradient resultant');
axis([-1 1 -1 1]); axis square; axis off;
text(1,0,'Target');
text(0,-1,'Target-Hand ');
text(-1,0,'Hand');
text(0,1,'Target+Hand');
```

As you can see, the neuron is tuned to the location of the target relative to the eyes. Now perform the same analysis for the idealized neurons that codes for;

Initial hand location in eye coordinates,

$$f = e^{-\left(\frac{H^2}{4}\right)}$$

Target location and initial hand location in eye coordinates,

$$f = e^{-\left(\frac{T^2}{2}+\frac{H^2}{8}\right)}$$

Target location in eye and hand coordinates,

$$f = e^{-\left(\frac{T^2}{4}+\frac{(T-H)^2}{4}\right)}$$

Difference vector in eye coordinates (target – hand),

$$f = e^{-\left(\frac{(T-H)^2}{2}\right)}$$

Examine the various response curves, see how tuning curves shift, as well as the direction of the resultant gradient vector.

Below you see a figure taken from Buneo et al (2002), plotting response fields and gradient vectors from neurons in the parietal reach region and area 5. Compare to the idealized response fields, and examine the difference between area 5 and PRR.

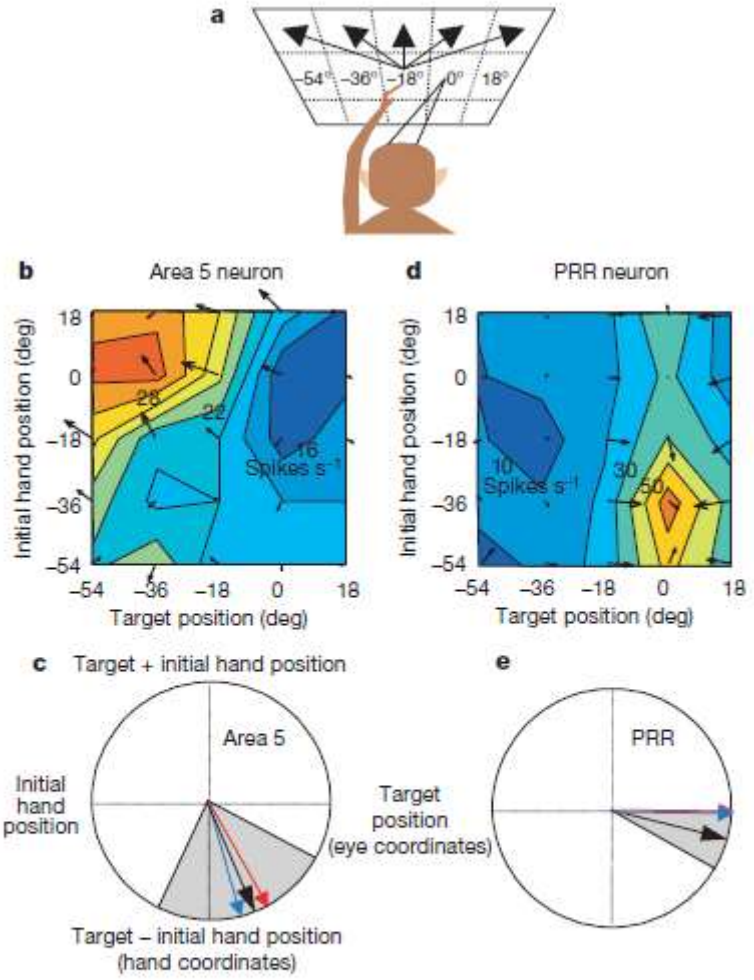


Figure 5 Results from the second experiment. **a**, Task schematic. **b**, Contour plot and gradient (black vectors) for one area 5 neuron. **c**, Resultant of the gradient in **b** (blue vector), population resultant of 15 single cell resultants (black vector), and gradient resultant for the idealized neuron in Fig. 4c (red vector). The shaded region indicates the 95% confidence interval for the population resultant[27]. **d**, Contour plot and gradient for a PRR neuron. **e**, Resultant of the gradient in **d** (blue vector), resultant of 17 PRR single cell resultants (black vector), and gradient resultant for the idealized neuron in Fig. 4b (red

Behavioral reference frames

Beurze et al. (2006) studied the behavioral reference frames in humans using a similar paradigm as Buneo et al. Subjects had to point to target locations with different directions of gaze and initial hand positions (see Figure 3). The authors measured the pointing errors as a function of these variables. Subjects were tested in two conditions, the unseen-hand condition and the seen-hand condition. Let's have a look at the their data, starting with the unseen hand condition. First load the data.

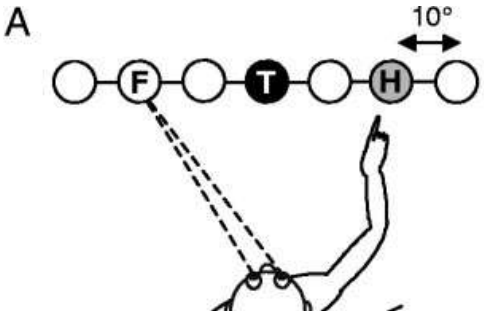


Figure 3: Paradigm Beurze et al. (2006)

```
load DataBeurze UnSeenHand
```

We first plot the data as a function of eye position.

```
plot(UnSeenHand.EyeReBody,UnSeenHand.DataEyeReBody,'-o')
legend('-30','-20','-10','0','10','20','30'), axis([-40 40 -4 4])
xlabel('Eye Position (deg)'); ylabel('Pointing error (deg)');
```

You will note the separate response curves for the different locations of the target. For example, the pointing error that occurs for a central target when gaze is deviated 10° to the right is about the same as the error for the pointing target at 20° with gaze directed 30° to the right. This suggest that error depends on the eye displacement relative to the targets rather than on gaze direction per se. To test this, we reorganize the data, and plot the error as a function of eye position relative to the target.

```
plot(UnSeenHand.EyeReTarget,UnSeenHand.DataEyeReTarget,'-o')
legend('-30','-20','-10','0','10','20','30'), axis([-40 40 -4 4])
xlabel('Eye re Target (deg)'); ylabel('Pointing error (deg)');
```

As shown, this rearranges the data into virtually a single response curve for all seven (body-fixed) target locations.

Next we can perform the same analysis for the effects of initial hand position.

```
plot(UnSeenHand.HandReBody,UnSeenHand.DataHandReBody,'-o')
legend('-30','-20','-10','0','10','20','30'), axis([-40 40 -4 4]);
xlabel('Hand Position (deg)'); ylabel('Pointing error (deg)');
```

Again, there are separate response curves for the various body-fixed targets. Note that the discontinuities in the curves here are due to the task restriction that initial hand position could not be the same as the target position. We can also plot the data as a function of hand displacement relative the target.

```
plot(UnSeenHand.HandReTarget,UnSeenHand.DataHandReTarget,'-o')
legend('-30','-20','-10','0','10','20','30'), axis([-40 40 -4 4]);
xlabel('Hand re Target (deg)'); ylabel('Pointing error (deg)');
```

Again, the data points collapse into a similar error pattern for all targets, which clearly indicates that the location of the target relative to the hand also has an effect on the pointing error.

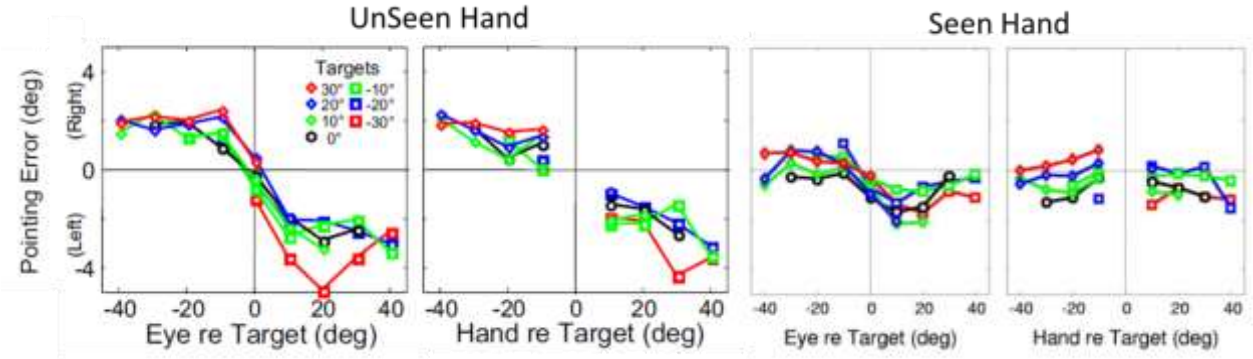
The Unseen Hand results indicate that the pointing errors can be well described as a function of either the eye- or hand-centered location of the target or both. If these results are to be explained within the eye-centered integration scheme (as in Buneo et al. 2002), they imply that the putative proprioceptive hand location signals are transformed into eye-centered coordinates in this condition.

Because it is unlikely that this transformation operates flawlessly, it can be expected to add noise to the system. Accordingly, the neural computations for eye-centered hand-target integration may be more accurate if this transformation is bypassed or assisted by providing visual information about initial hand position at the moment a movement plan is being constructed. Now have a look at the data of the SeenHand condition to see whether this is the case by using the routines above. What is would you expect to happen? [LS: I guess you want an increase in precision?]

```
load DataBeurze SeenHand
```

As a common reference frame is required to specify a displacement vector, these results suggest that an eye-centered mechanism is involved in integrating target and hand position in programming displacements vectors. In Beurze et al. (2006) it is modeled and discussed how simple gain elements modulating the eye-centered target and hand position signals can account for these results.

Below you find the experimental results of Beurze et al., as taken from the paper.



Optimal weighting of reference frames in reach planning

Mcguire and Sabes (2009) further build on the above observations and proposed a model in which the statistical properties of sensory signals and their transformations determine how these target and hand position signals are used in computing the movement difference vector. Their model provides a possible account for the errors observed in Beurze et al. (2006). Let's have a look at their model and implement the necessary steps in Matlab.

The model is given in the figure below. The model begins with sensory inputs signaling target location, initial hand position and gaze location. They are modeled as Gaussian likelihoods of true location given the sensory input, with likelihood variance reflecting the reliability of the sensory modality. Visual signals arrive in retinotopic coordinates and proprioceptive signals arrive in a body-centered coordinates. Each available signal is then transformed into the non-native reference frame, which can be approximated simply by adding or subtracting the gaze location: or in probabilistic terms by convolving their distributions. Because the internal estimate of gaze location is also uncertain, this transformation adds variability to the signals. When both sensory modalities are available, the native and transformed signals are integrated in both reference frame representations (so the target representation exists both in the proprioceptive and in the visual domain, as does the hand location). Displacement vectors are then computed in each representation by convolving the target and initial hand distributions (subtraction). Because the sensory transformation adds variability, each spatial variable is more reliably represented in one or the other of these representations depending on the availability and reliability of the relevant visual and proprioceptive signals.

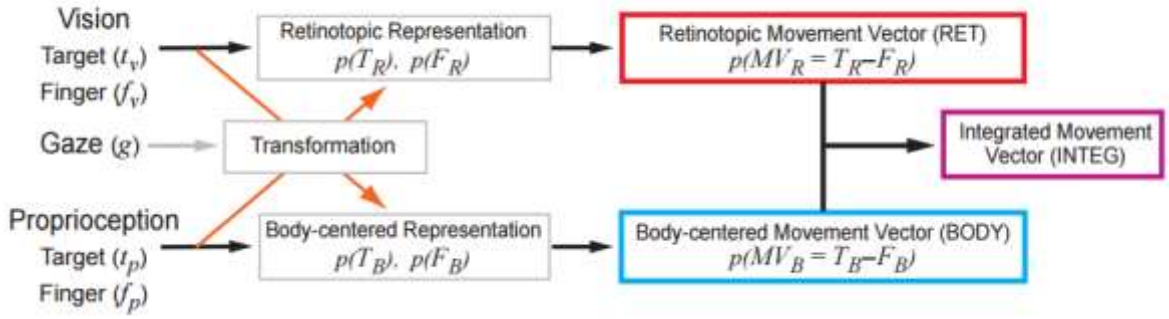


Figure S6: Movement vector planning in multiple coordinate frames. a) Flow of information for movement planning. From left to right: Sensory information is represented in native coordinate frame (black arrows) and transformed into non-native representation (orange arrows). Combined estimates are used to compute a movement vector in each coordinate frame. Final movement plan may be read out from a single representation (RET, BODY) or using both representations (INTEG).

The model defines 5 potential sensory representations, vision of the target (tv), vision of the finger (fv), proprioception of the target (tp), proprioception of the finger (fp), and gaze position(g). All are described as Gaussian probability distributions centered on the true locations: Tv, Fv, Tp, Fp, G, respectively. Thus $p(tv|Tv) \sim N(Tv,\sigma^2)$ etc.

When a signal is unavailable, the likelihood is the uniform distribution. The likelihood represents variability in a sensory signal x given the true location X. The computations in the model, however, depend on uncertainty in X given x, that is, on the posterior distributions $p(X \mid x)$, which can be computed using Bayes' rule $p(X \mid x) \propto p(x \mid X)\, p(X)$. Here $p(X)$ represents the prior, which basically is flat (but see below), so that the posterior is proportional to the likelihood.

The model performs only two operations: signal **integration** and **addition**.

Integration: $p(X \mid x1,x2) \propto p(X \mid x1)\, p(X \mid x2)\, p(X)$

Addition: $p(Z \mid y,x) \propto p(X \mid x) \otimes p(Y \mid y)$ ($\otimes$=convolution)

Let's now implement the model in Matlab. The function normpdf(x,MU,SIGMA) returns the probability density function of the normal distribution with mean MU and standard deviation SIGMA, evaluated at the values in x. Thus, for example,

```
x=[-200:1:200]; % define location space
g =-20;       % eye position re body, set to -20
sig_g=5;      % eye position re body set to 5
P_g = normpdf(x,g,sig_g);
```

Do the same for P_tv, P_fv, P_tp, P_fp. Choose your own values of MU and SIGMA.

The model first builds internal representations of fingertip and target location in both retinotopic in body-centered reference frames, requiring transformations with gaze. For example, computing the body representation of the target, the visual signal tv must be transformed:

$$p(T_B \mid tv, g) = p(T_R \mid tv) \otimes p(G \mid g)$$

Likewise, computing the retinotopic representation of the target, the proprioceptive signal *tp* must be transformed

$$p(T_R \mid tp, g) = p(T_B \mid tp) \otimes p(-G \mid g).$$

In Matlab,

```
P_tv_BODY=conv(P_tv, P_g,'same');

P_tp_RET=conv(P_tp, P_g(end:-1:1),'same'); %note the trick, P_g is flipped
in zero to operate the convolution as a subtraction
```

Parallel transformations are made for the visual and proprioceptive signals of the fingertip, which you can add yourselves.

Now the model has two independent estimates of the same variable, which can be integrated, for example,

$$p(T_R \mid tp, tv, g) \propto p(T_R \mid tv)\, p(T_R \mid tp, g)$$

In Matlab,

```
P_target_RET=P_tv.*P_tp_RET; %integration
P_target_RET=P_target_RET/sum(P_target_RET); %normalization
```

Now do the same for the other three variables.

The model next computes retinotopic and body-centered representations of the desired displacement vector ("target minus hand"),

$$p(MV_R \mid tp, tv, fv, fp, g) \propto p(T_R \mid tv, tp, g) \otimes p(F_R \mid fv, fp, g)$$

$$p(MV_B \mid tp, tv, fv, fp, g) \propto p(T_B \mid tv, tp, g) \otimes p(F_B \mid fv, fp, g)$$

In Matlab,

```
% compute displacement vectors
P_MV_RET=conv(P_target_RET, P_finger_RET(end:-1:1),'same');
P_MV_BODY=conv(P_target_BODY, P_finger_BODY(end:-1:1),'same');
```

Finally, the model selects a planned displacement vector using one of three readout schemes, either the MV_RET or MB_BODY posteriors, or an integrated representation of these two posteriors, assuming they are independent:

$$p(MV_{INTEG} \mid tp, tv, fv, fp, g) \propto\ p(MV_R \mid tp, tv, fv, fp, g)\, p(MV_B \mid tp, tv, fv, fp, g)$$

In Matlab,

```
P_MV_INTEG=P_MV_RET.*P_MV_BODY; %integration
P_MV_INTEG=P_MV_INTEG/sum(P_MV_INTEG); %normalization
```

The three readouts can be taken as maximum a posterior estimates, which in Matlab is as follows:

```
MV_RET=x(find(P_MV_RET==max(P_MV_RET)));
MV_BODY=x(find(P_MV_BODY==max(P_MV_BODY)));
MV_INTEG=x(find(P_MV_INTEG==max(P_MV_INTEG)));
```

Examine the computations of the model by plotting the various probability distributions. Also, play around with input signals and their variances.

- What happens in the visual variance depends on the location relative to gaze?
- When happens if one of the input signals is unavailable?
- Does the model make systematic errors? Why or why not?

The final component of the model is a bias in the transformation of the target location between reference frames. This component posits that the internal estimate of gaze direction used to transform target location is biased toward the target. The bias can be modeled as a Bayesian prior on gaze location centered on the target: $p(G) \sim N(t,\sigma^2)$. Because the transformation consists of adding or subtracting gaze location, depending on the direction of the transformation, the prior effectively biases the transformed target estimate either away from or toward the direction of gaze. Native (untransformed) target representations remain unbiased. The prior does not affect the transformation of finger locations.

In Matlab, add the following code:

```
gp=normpdf(x, tp, sig_gp); %prior on gaze
P_g_gp=P_g.*P_gp; P_g_gp=P_g_gp/sum(P_g_gp); %gaze biased to target
P_tp_RET=conv(P_tp, P_g_gp(end:-1:1),'same'); %transformed tp
P_tv_BODY=conv(P_tv, P_g_gp,'same'); %transformed tv
```

Examine the effect of the transformation bias on the readout of the model.

Finally, a matlab version of the model is given by integ.m. You can play with this model either in 'single trial' or 'multiple trial' version. Can you make the model capture the data of Beurze et al?