Motion Model Estimation

The task given was to calculate the alpha parameters based on some experiments we performed.

```
clear;
```

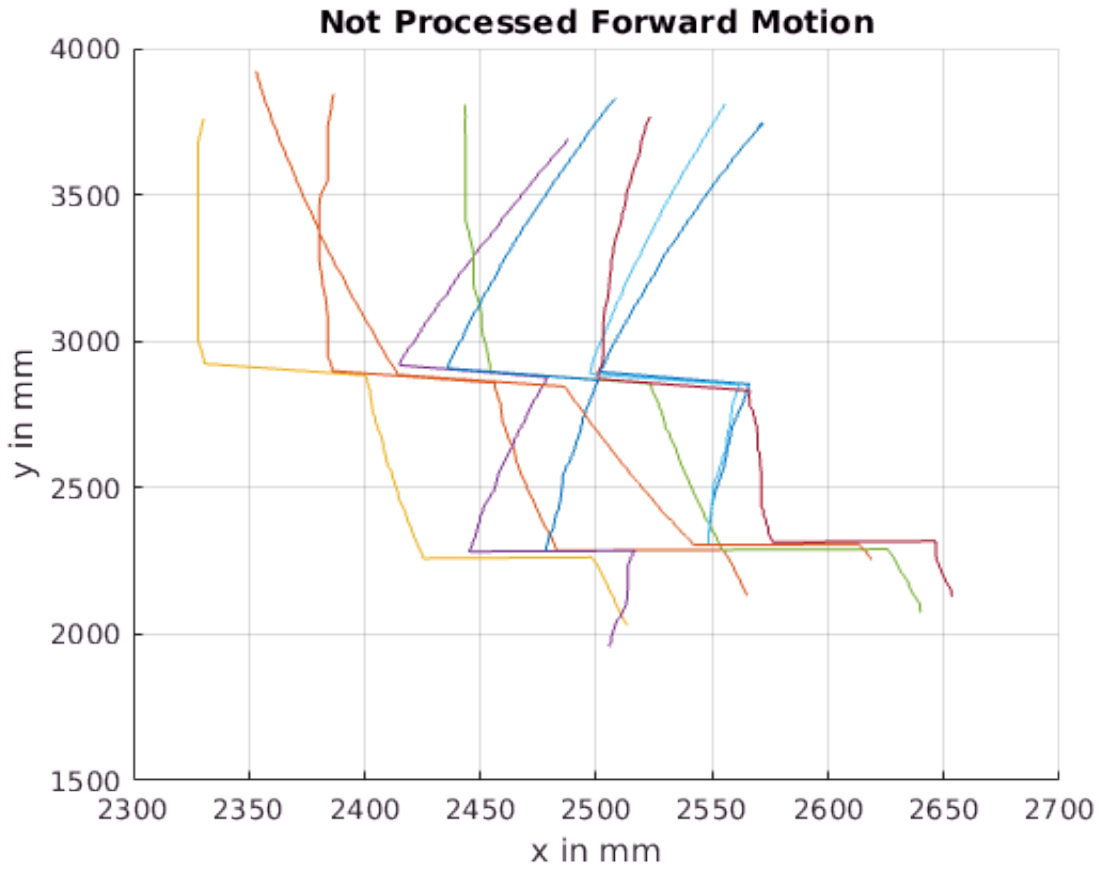
Read all log files

```
forwardFiles = dir('./logs/forward*.log');
% Forward motion
for iFile = 1:length(forwardFiles)
    read = importdata(sprintf('./logs/%s',forwardFiles(iFile).name),...
        ' ', 0);
    forward(iFile).time = read(:,1);
    forward(iFile).x = read(:,2);
    forward(iFile).y = read(:,3);
    forward(iFile).orientation = read(:,4);
    forward(iFile).duration = forward(iFile).time(end)...
        - forward(iFile).time(1);
end

% Right turn
read = importdata('./logs/right_turn.log',' ',0);
right.time = read(:,1);
right.x = read(:,2);
right.y = read(:,3);
right.orientation = read(:,4);
right.duration = right.time(end) - right.time(1);

% left turn
read = importdata('./logs/left_turn.log',' ',0);
left.time = read(:,1);
left.x = read(:,2);
left.y = read(:,3);
left.orientation = read(:,4);
left.duration = left.time(end) - left.time(1);
```

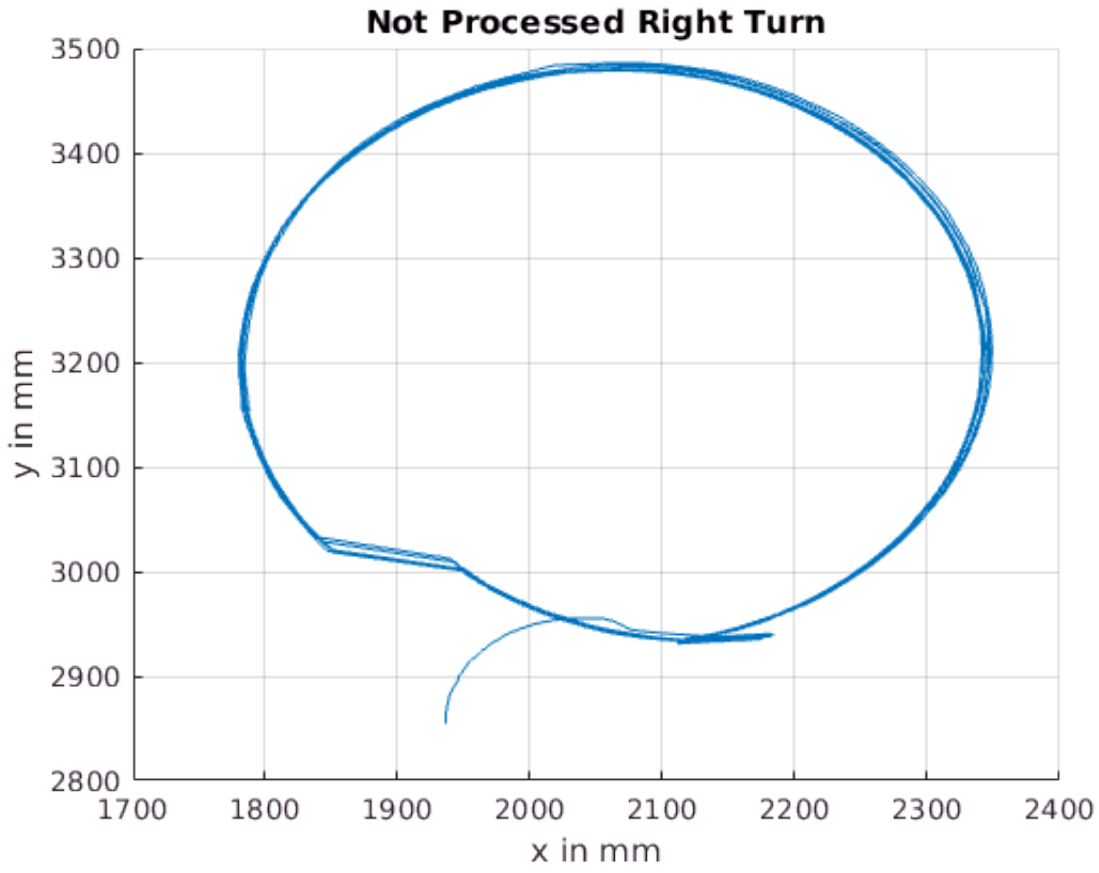
Motion Vizualizations

Plot forward motion

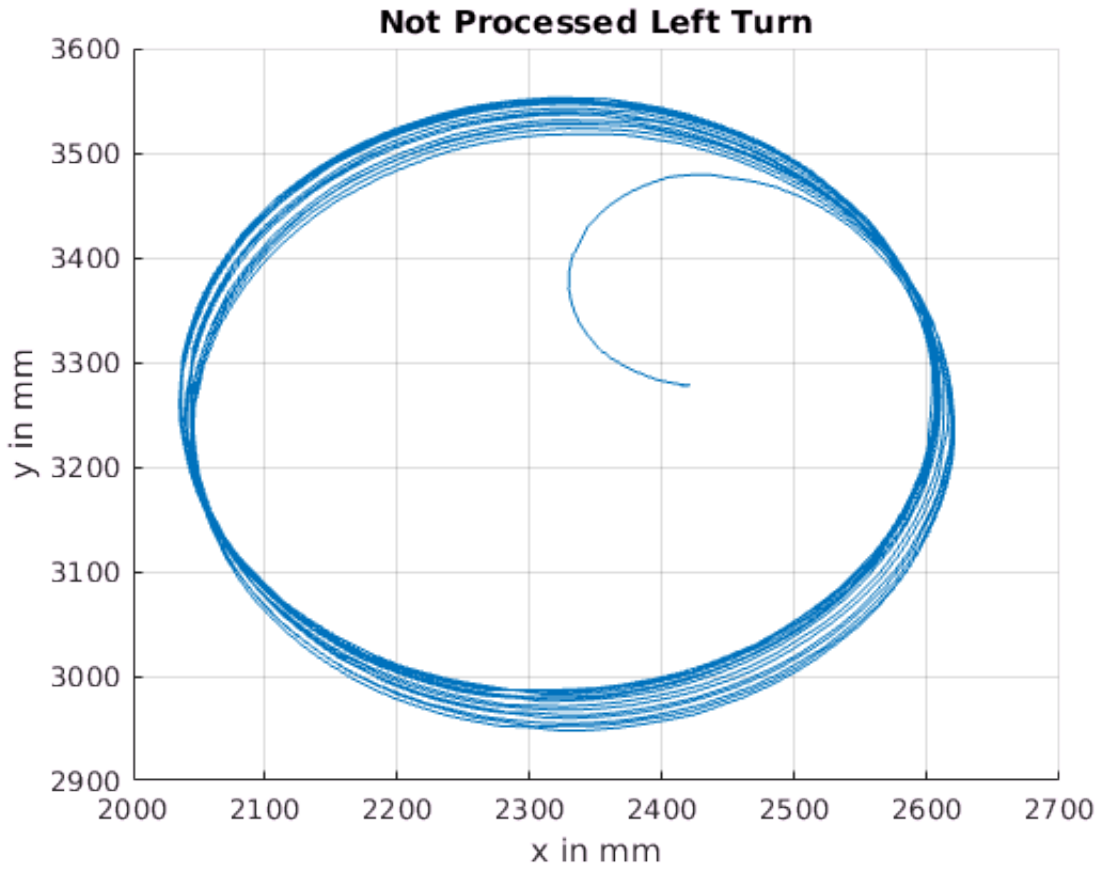
```
figure(1);clf;hold on;
for i=1:length(forward)
    plot(forward(i).x, forward(i).y);
end
title('Not Processed Forward Motion');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```




```
% Plot right turn
figure(2);clf;hold on;
plot(right.x, right.y);
title('Not Processed Right Turn');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```




```
% Plot left turn
figure(3); clf; hold on;
plot(left.x, left.y);
title('Not Processed Left Turn');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```



Split left turns into single experiments

```
sequenceDuration = 2000;%round(left.duration/20);
index = 1;
iExperiment = 1;
for sample = 1:length(left.time)
    if left.time(sample) >= left.time(1) + iExperiment * sequenceDuration
        index = 1;
        iExperiment = iExperiment + 1;
    end
    leftExperiment(iExperiment).time(index) = left.time(sample);
    leftExperiment(iExperiment).x(index) = left.x(sample);
    leftExperiment(iExperiment).y(index) = left.y(sample);
    leftExperiment(iExperiment).orientation(index) =...
        left.orientation(sample);
    index = index + 1;
end
```

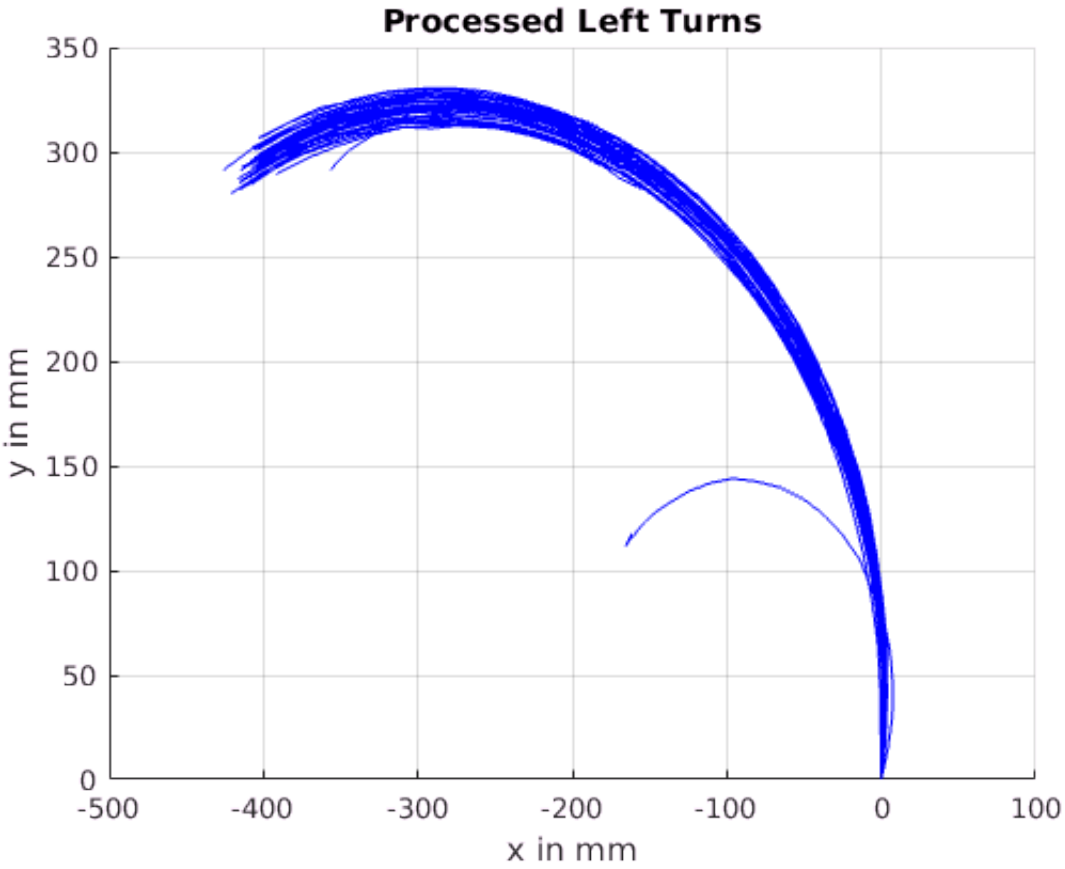
Make every experiment start from the same start pose

```
for i = 1:length(leftExperiment)
    % Get orientation
    offsetX = leftExperiment(i).x(1);
    offsetY = leftExperiment(i).y(1);
    offsetOrientation = leftExperiment(i).orientation(1);

    for j = 1:length(leftExperiment(i).time)
```

```
        % translate
        x = leftExperiment(i).x(j) - offsetX;
        y = leftExperiment(i).y(j) - offsetY;
        orientation = leftExperiment(i).orientation(j) - offsetOrientation;

        % rotate
        leftExperiment(i).x(j) = cos(-offsetOrientation+pi)*x...
            - sin(-offsetOrientation+pi)*y;
        leftExperiment(i).y(j) = sin(-offsetOrientation+pi)*x...
            + cos(-offsetOrientation+pi)*y;
        leftExperiment(i).orientation(j) = orientation;
    end
end
```

Plot Left turns

```
figure(4);clf;hold on;
for experiment = 1:length(leftExperiment)
    plot(leftExperiment(experiment).x, leftExperiment(experiment).y,'b-');
end
title('Processed Left Turns');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```



Split right turns into single experiments

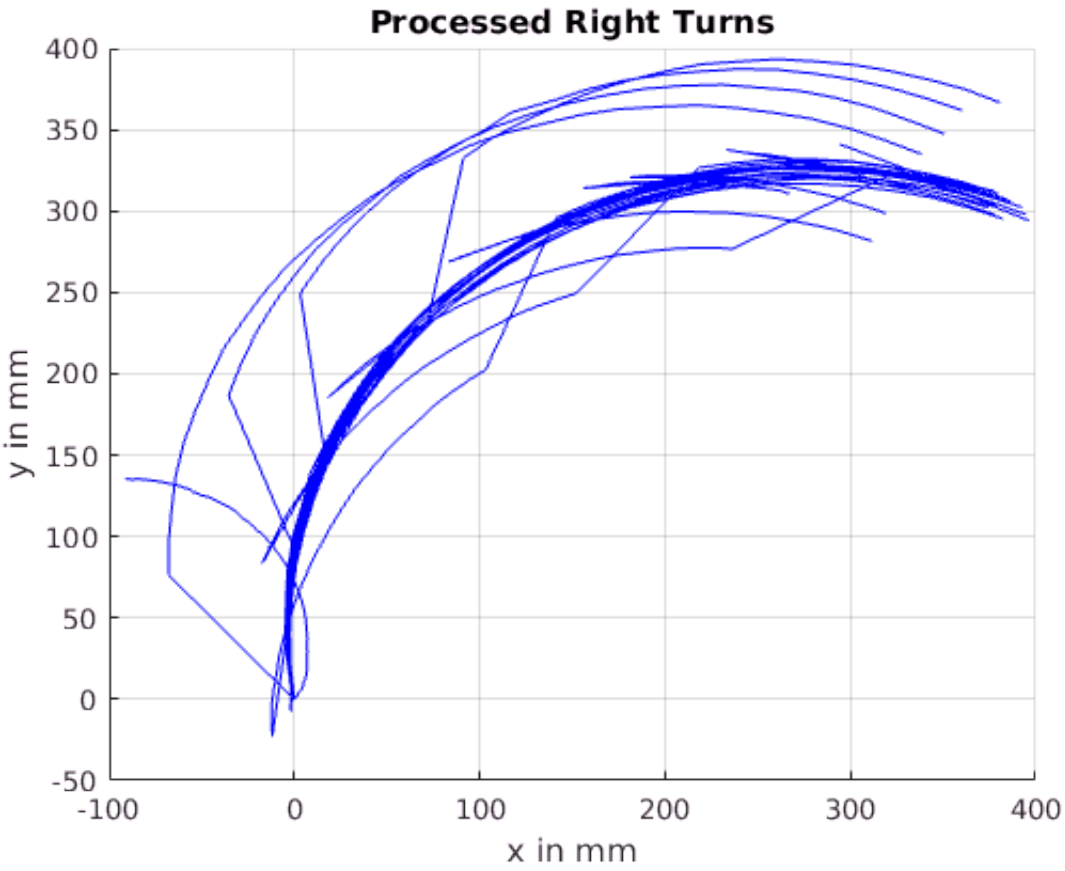
```
index = 1;
iExperiment = 1;
for sample = 1:length(right.time)
    if right.time(sample) >= right.time(1) + iExperiment * sequenceDuration
        index = 1;
        iExperiment = iExperiment + 1;
    end
    rightExperiment(iExperiment).time(index) = right.time(sample);
    rightExperiment(iExperiment).x(index) = right.x(sample);
    rightExperiment(iExperiment).y(index) = right.y(sample);
    rightExperiment(iExperiment).orientation(index) =...
        right.orientation(sample);
    index = index + 1;
end
```

Make every experiment start from the same start pose

```
for i = 1:length(rightExperiment)
    % Get orientation
    offsetX = rightExperiment(i).x(1);
    offsetY = rightExperiment(i).y(1);
    offsetOrientation = rightExperiment(i).orientation(1);

    for j = 1:length(rightExperiment(i).time)
        % translate
        x = rightExperiment(i).x(j) - offsetX;
        y = rightExperiment(i).y(j) - offsetY;
        orientation = ...
            rightExperiment(i).orientation(j) - offsetOrientation;

        % rotate
        rightExperiment(i).x(j) = cos(-offsetOrientation+pi)*x...
            - sin(-offsetOrientation+pi)*y;
        rightExperiment(i).y(j) = sin(-offsetOrientation+pi)*x...
            + cos(-offsetOrientation+pi)*y;
        leftExperiment(i).orientation(j) = orientation;
    end
end
```

Plot Right turns

```
figure(5);clf;hold on;
for experiment = 1:length(rightExperiment)
    plot(rightExperiment(experiment).x, rightExperiment(experiment).y,'b-');
end
title('Processed Right Turns');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```



Plot Forward Motion

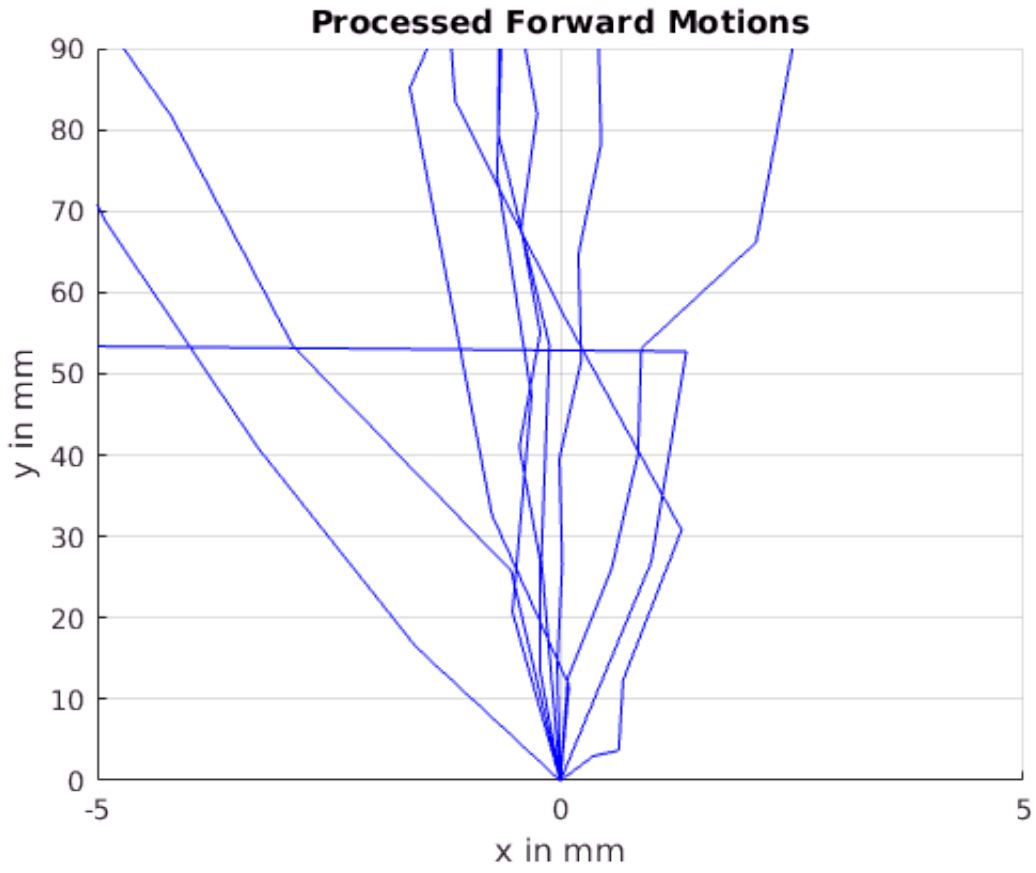
Split forwards into single experiments

```
index = 1;
iExperiment = 0;
for i = 1:length(forward)
    iExperiment = iExperiment + 1;
    index = 1;
    for sample = 1:length(forward(i).time)
        if forward(i).time(sample) >=...
                forward(i).time(1) + iExperiment * sequenceDuration
            index = 1;
            iExperiment = iExperiment + 1;
        end
        forwardExperiment(iExperiment).time(index) =...
            forward(i).time(sample);
        forwardExperiment(iExperiment).x(index) =...
            forward(i).x(sample);
        forwardExperiment(iExperiment).y(index) =...
            forward(i).y(sample);
        forwardExperiment(iExperiment).orientation(index) =...
            forward(i).orientation(sample);
        index = index + 1;
    end
end
```

```
% Make every experiment start from the same start pose
for i = 1:length(forwardExperiment)
    % Get orientation
    offsetX = forwardExperiment(i).x(1);
    offsetY = forwardExperiment(i).y(1);
    offsetOrientation = ...
        forwardExperiment(i).orientation(1);

    for k = 1:length(forwardExperiment(i).time)
        % translate
        x = forwardExperiment(i).x(k) - offsetX;
        y = forwardExperiment(i).y(k) - offsetY;
        orientation = ...
            forwardExperiment(i).orientation(k)...
            - offsetOrientation;

        % rotate
        forwardExperiment(i).x(k) =...
            cos(-offsetOrientation+pi)*x...
            - sin(-offsetOrientation+pi)*y;
        forwardExperiment(i).y(k) =...
            sin(-offsetOrientation+pi)*x...
            + cos(-offsetOrientation+pi)*y;
        forwardExperiment(i).orientation(k) = orientation;
    end
end
```

Plot Forward Motion

```
figure(6);clf;hold on;
for i = 1:length(forwardExperiment)
        plot(forwardExperiment(i).x,...
            forwardExperiment(i).y, 'b-');

end
axis([-5 5 0 90]);
title('Processed Forward Motions');
xlabel('x in mm');
ylabel('y in mm');
grid on;
```



Calculate mle for movements

Forward

```
%concatenatedForward = concatenateValues(forwardExperiment);
forwardExperiment = calculateVelocities(forwardExperiment);
for i = 1:length(forwardExperiment)
    if i==3
        forwardExperiment(i).pV = 0;
        forwardExperiment(i).pW = 0;
        continue;
    end
    forwardExperiment(i).pV = mle(forwardExperiment(i).v,'dist','normal');
    forwardExperiment(i).pW = mle(forwardExperiment(i).w,'dist','normal');
end

% Right
%concatenatedRight = concatenateValues(rightExperiment);
rightExperiment = calculateVelocities(rightExperiment);
for i = 1:length(rightExperiment)
    rightExperiment(i).pV = mle(rightExperiment(i).v,'dist','normal');
    rightExperiment(i).pW = mle(rightExperiment(i).w,'dist','normal');
end


% Left
%concatenatedLeft = concatenateValues(leftExperiment);
leftExperiment = calculateVelocities(leftExperiment);
for i=1:length(leftExperiment)
```

```
    leftExperiment(i).pV = mle(leftExperiment(i).v,'dist','normal');
    leftExperiment(i).pW = mle(leftExperiment(i).w,'dist','normal');
end
```

Find Alphas

Alpha calculation is still missing as for I am not entirely sure how to do it

```
%solve([leftExperiment(1).v'; leftExperiment(1).w']*x,x);

% [alpha1, alpha2] = computeAlpha(forwardExperiment.v',...
%     forwardExperiment.w',forwardExperiment.pV');
%
% [alpha3, alpha4] = computeAlpha([left_v(:,1); right_v(:,1)], ...
%                     [left_w(:,1); right_w(:,1)], ...
%                     [left_w(:,2); right_w(:,2)]);
% alpha5 = 1.0;
% alpha6 = 1.0;
```

Compute Motion Model

Computation of motion model has to be done similar to this for each observed pose. Like this, for every observation a notion about how likely it is will be calculated.

```
vErr = [];
wErr = [];
gammaErr = [];
for i=1:length(leftExperiment)
    for j=1:length(leftExperiment(i).time)-1
        x = leftExperiment(i).x(j);
        y = leftExperiment(i).y(j);
        theta = leftExperiment(i).orientation(j);
        xNew = leftExperiment(i).x(j+1);
        yNew = leftExperiment(i).y(j+1);
        thetaNew = leftExperiment(i).orientation(j+1);
        deltaT = leftExperiment(i).time(j+1) - leftExperiment(i).time(j);
        v = sqrt((xNew - x)^2 + (yNew - y)^2)/deltaT;
        w = (thetaNew - theta)/deltaT;
        [a b c]= motionModel(x,y,theta,v,w,xNew,yNew,thetaNew...
            ,deltaT);
        vErr = [vErr a];
        wErr = [wErr b];
        gammaErr = [gammaErr c];
    end
end
```