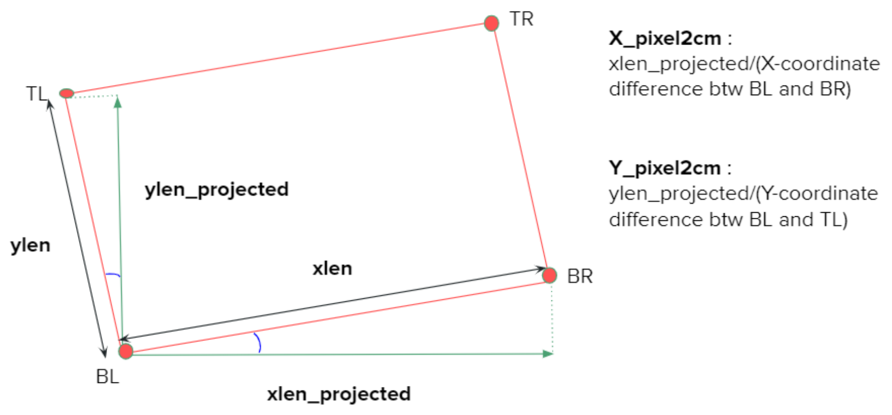
Cat Odor Analysis (Run_CatOdor.m)

1. **Main_CatOdor.m:** Calls all the functions and carries out the complete analysis. It first loads the coordinates and sets the values for various parameters. This is followed by quality checking, coordinate correction, diagonal measurement and ratio measurement. Next, boundaries are created in the experimental box to define the safe and the danger zones. Different types of plots are then created to represent the location of the mouse at different instances. Eventually, the risk_function.m is called to carry out the stretching and the to-fro behavior analysis.
2. **CheckQuality.m:** It calculates the percentage of frames below a quality threshold value (0.95) using the likelihood values generated by DLC for each coordinate for each frame. Higher the likelihood value (i.e., closer to 1), the more accurate is the DLC prediction. The function calculates these percentages individually for all the coordinates, and then cumulatively as well.
3. **CorrectCoordinates.m:** It first calculates the mean of the bottom left, bottom right and the top left corners of the experimental box. It is possible that the box may not be placed in a perfect horizontal position with respect to the camera, and therefore may appear at an angle with respect to the horizontal.
   The x coordinate difference between the bottom left and the bottom right coordinates is compared with the projected length of the box in the x direction to get the pixel-to-centimeter conversion rate in the x direction (*x_pixel2cm*). Similarly, the y coordinate difference between the top left and the bottom left coordinates is compared with the projected breadth of the box in the y direction to get the pixel-to centimeter conversion rate in the y direction (*y_pixel2cm*).
   Now all the coordinates are first re-referenced with the bottom left corner of the box as the origin and the coordinates' x and y components are multiplied with the *x_pixel2cm* and *y_pixel2cm* respectively to get the new, corrected coordinates.


TR
TL
ylen_projected
ylen
xlen
BR
BL
xlen_projected
X_pixel2cm :
xlen_projected/(X-coordinate difference btw BL and BR)
Y_pixel2cm :
ylen_projected/(Y-coordinate difference btw BL and TL)

4. **Diagonal.m:** It just calculates the diagonal length of the box, from the top-left to the bottom right using the new, corrected coordinates.
5. **Ratio.m:** It calculates the ratio of the distance covered by the nose to the distance covered by the tailbase. This could be used as an indicator of sniffing.

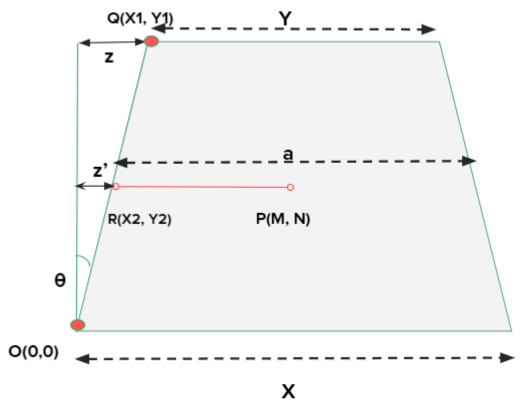
6. **densityplot.m:** It is used to create a density plot using the coordinates of the mouse location from multiple frames.
7. **Dotplot.m:** It is used to create a dot plot to indicate the coordinate locations of any particular marker in a specified number of frames.
8. **Lineplot.m:** It is used to represent the path of the animal, color coded by the likelihood values of the predicted DLC coordinates. Coordinates with likelihood values below 0.5 are colored red, those below 0.8 are color coded blue and the remaining ones are color coded cyan.
9. **Distance.m:** It calculates the total distance traveled by the mouse in a specified time frame, by using any particular marker.
10. **Length.m:** It calculates the individual lengths between the 9 consecutive markers, and these 8 values are added to generate the length of the animal in all the frames.
11. **Speed.m:** It calculates the speed of the animal for all the frames.
12. **Risk_function.m:** This function is used to compute the stretching behavior and the to-fro motion of the animal. In general, the analysis is performed in the period from the start of the experiment till the moment the fabric is moved out of the danger zone.
    - The amount of time spent in the safe and danger zones are calculated, by observing the location of the nose with respect to the defined boundaries.
    - The average length of the animal in the entire analysis period is calculated.
    - The average length of the animal when present in the danger zone, safe zone and the middle zone are respectively calculated.
    - An upper threshold is defined by the sum of the mean and standard deviation of the lengths (in the case of cat experiments, this upper threshold was derived from the basal videos). The average length of the animal above this threshold and the average length of the animal above this threshold when present in the danger zone are computed.
    - During these length calculations, in order to identify the position of the mouse, the coordinates of the nose are used for analysis.
    - A graph of the length of the animal versus the frames is plotted. This graph is color coded to also indicate if the animal was present in the danger, middle or the safe zone.
    - A plot is created in which the mouse posture is plotted for all the frames when its length is greater than the upper threshold. Specifically, only the nose, bodycentre and the tailbase coordinates are used for this posture plotting. The number of frames with the stretched length and stretched length in the danger zone are computed for before and after the beginning of the cat experiment.
    - In order to understand to-fro motion of the mouse, two variables, *pos_left* and *pos_right* are created. For a particular frame number, the value of *pos_left* is 1 if the bodycentre of the animal is within the danger zone, else 0. Similarly, for *pos_right*.
    - The frame numbers at which the animal leaves the danger zone, enters the safe zone, leaves the safe zone and re-enters the danger zone are identified and stored in an index variable. A plot showing the x-coordinate of the animal versus

frames is plotted and it indicates the moments when the animal leaves the danger zone and re-enters it, to highlight periods of to-fro behavior.

- The movement of the animal from the danger zone to the safe zone is defined as an escape response and the movement from the safe zone to the danger zone is defined as an approach response. The speed of the animal during each escape and approach event are also computed.
- All the escape responses from the moment the animal leaves the danger zone till the moment the animal enters the safe zone, are then plotted.
- It returns a structure variable that stores all the details for all the responses.

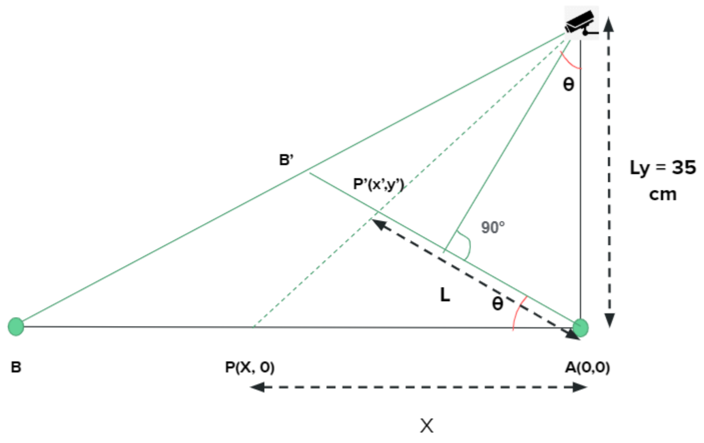
<u>**LOOM-SWEEP ANALYSIS (Run_Loom.m and Run_Sweep.m):**</u>

1. **Main_LoomSweep.m:** Calls all the functions and carries out the complete analysis. It first loads the coordinates and sets the values for various parameters. This is followed by the transformation of coordinates, creating of the location variable and eventually understanding the pattern of escape responses.
2. **TransformCoordinates.m:** It is used to convert the trapezium based coordinates to real world coordinates given the dimensions of the square experimental area and the height at which the camera was installed. The algorithm for the coordinate transformation is as follows:
    - The aim is to convert any image coordinate P(M,N) to real world P(m,n) coordinates.
    - <u>X-coordinate transformation:</u> The idea is that in the real world, Y=X. To achieve this from the image, we have to add (2*z) to Y, to make it equal to X.
        - ➢ So to make Y/n equal to X/n, for any value of n, (2*z/n) will have to be added.
        - ➢ The equation of the line connecting OQ can be written as y = (Y1/X1)*x. For P(M,N), the corresponding point on the line will be R(X2,Y2), with Y2 = N and X2 = (X1/Y1)*N.
        - ➢ The horizontal line passing through R and P and covering the length of the plane has a length ‘a’. To convert ‘a/n’ to X/n, (2*z’/n) will have to be added.
        - ➢ Length of RP in the image = **M - X2**
          = **M - (X1/Y1)*N**
        - ➢ Considering RP as ‘a/n’, the corresponding value of ‘n’ will be:
          n = a/[M - (X1/Y1)*N]
        - ➢ Value of RP in real world can then be given by:
          RP^{rw} = Length of RP in image + (2*z’/n)
          = [M - (X1/Y1)*N] + (2*z’)*[M - (X1/Y1)*N]/a
        - ➢ Pixel-to-cm rate can be calculated by measuring the length of the front edge of the field in image and equating the length to 50cm.
        - ➢ The real x-coordinate, **m = Pixel-to-cm * RP^{rw}**



- <u>Y-coordinate transformation:</u> The camera is tilted at an angle θ at a height 35 cm. From the side view, we can imagine that the real world field AB is captured as AB' in the image. Here, the point P' corresponds to any point on the image plane. However, its coordinates (x',y') are with respect to the side view. In the real world, this point would be given by P(X,0) where X would represent the distance of the point from the front edge. Assume that the point P'(x',y') lies at a distance of L from the front edge.

  - ➢ The equation of the line connecting PP' can be given by:
    $$\mathbf{x/X + y/L_y = 1}$$
  - ➢ By geometric analysis, we can calculate:
    $$\mathbf{x' = L*\cos(\theta) \text{ and } y' = L*\sin(\theta)}$$
  - ➢ By plugging x' and y' in the equation of the line, we can calculate X as:
    $$\mathbf{X = L*\cos(\theta)*L_y/(L_y - L*\sin(\theta))}$$
  - ➢ We assume that if the camera was placed at an upright position, then the pixel-to-cm conversion ratios would have been equal in the x and in the y directions. We use this to estimate the angle θ. In the above equation, we can use X = 50 cm, and L = AB' * pixel-to-cm, and calculate the value of θ.



3. **Freezing.m:** To analyze the freezing responses, we created a freeze detection algorithm that allows the identification of all the periods of stationarity. A small square is first constructed around the location variable (loc) for each frame, and the distance of any edge from loc is given by a new parameter *max_dist*. Here, we used a value of 0.25 cm for *max_dist*. For every frame, we calculated the number of frames, starting from the current one, until which the animal's location coordinate was present within the defined square for the starting frame. A graph was constructed with time (frames) on the horizontal axis. The y-value corresponding to each frame on the x-axis indicated the total number of frames the animal stayed within that frame's boundary square. We defined another parameter, *min_peak_height*, and used a value of 9 for this. It indicates the minimum number of frames an animal must stay within a defined boundary for the response to be classified as a freezing response. The frame rates of all the looming and sweeping videos were around 30 fps, and a value of 9 for min_peak_height ensured that the minimum freezing duration was 300 ms. The period starting from the peak value till the frame the y-value first reduced to unity was defined as an instance of freezing behavior.
4. **Escape_Responses.m**: It performs the main classification of responses, as being an escape, or freezing or a mixture of the two. Several parameters are constructed to identify and further classify escape responses as fast escape, slow escape, attempted escape or no escape. It also calls the Freezing.m function to identify the nature of freezing. These responses are then plotted in terms of the trajectories, velocities and raster plots. It returns a structure variable that stores all the details of all the responses.
5. **DetectStartofEscape.m:** It is used to compute the beginning of an escape response. The algorithm uses a back traversal method. The peak velocity acquired by the animal during escape is identified. The iterator variable then travels backward, starting from the peak velocity frame, till the frame when the velocity first reaches a value of 5 cm/sec.