Documentation Tracker\_object

The object class defined as tracker implement the main entity used in the software to extrapolate animal position from the acquired frames.

The slot devoted to process the frames and track animal position requires the following inputs:

void tracker::start\_tracking(

cv::Mat \*resultROI //matrix to store the frame containing the detected object

cv::Mat \*threshROI //matrix to store the frame after the threshold is applied

cv::Mat \*frameROI //matrix containing the current frame to process

cv::Mat \*backROI //matrix to store the calculated background

cv::Mat \*foreROI //matrix to store the calculated foreground

cv::BackgroundSubtractor \*bgROI //object from the class BackgroundSubtractor implemented in opencv 3.0. It is perform the calculation of the foreground detection

int \*xposROI //integer used to store animal position coordinates

int \*yposROI //integer used to store animal position coordinates

int \*frame\_processedROI //integer used to keep track of the number of frames processed

NOTE: all the variable above are parsed to the public slot **start\_tracking** as pointers. This will reduce the computational load on the main thread which will have to handle many different tracking objects at the same time. If you have no requirement for online tracking or computation optimization just parse to the slot the variable itself, adjusting of course the syntax accordingly in the object body.

This class of objects relies on three main signals:

void send\_position(int, int, int);

Signal used to output the position of the animal (x,y) together with the area of the detected moving object (the animal) in the following order: x,y,size.

void send\_position\_conditioning(int, int, int);

This signal is basically the same as the above one, but it is used when you want to perform a conditioning session. This is convenient in case you want to perform several sessions, one after each other, alternating between conditioning and test sessions. The reason why we chose this approach is that we use an independent thread for handling the stimulus delivery during the conditioning session, hence the necessity of an additional output signal.

NOTE: you can toggle a Boolean in the public slot:

void conditioning (bool conditioning\_activated)

and keep track if you are performing a conditioning session or not.

void send\_clock(bool);

This signal is used to time the acquisition of new frames by the tracker. This is very important once you are implementing a multi-threads approach since you do not wanna generate a cue of frames waiting to be processed by the tracking-thread. This is mainly due to the computational time each frame needs to be processed. We solved this issue by using this signal to tell the image acquisition object when to send a new frame. When the object tracker receives a frame it emits a signal send\_clock (false), it processes the frame and once done it emits a new signal send\_clock (true). Only when a signal send\_clock (true) it is emitted a new frame will be send t the object for processing.

NOTE: this approach is unnecessary if you are doing offline processing of your videos.