Once the Inputs are taken from the User which are the Video footage and the Target image, the program starts 2 child processes lets name them as Parent Process(PP) the main one, Algorithm Process(AP) first child, Output Process (OP) second child. To ensure communication between them, 2 pipes are created in advance and let them be Parent-Algorithm Pipe (PA pipe), and Algorithm-Output Pipe (AO Pipe), as their name suggests the PA pipe is used for communication between Parent Process and Algorithm Process, same for AO Pipe.

Once all the required objects are created, the Parent Process reads the frames from the video footage sequentially and send them to the Algorithm process through the PA pipe, this is the only thing the parent process will be doing from now until all the frames are read.

The Algorithm Process has 2 functions, one function runs the Particle Swarm Optimization Algo, where it creates the no of particles (Every particle has a co-ordinate where it lies , a coordinate of its personal best result, its personal best score and a function to update its personal best variables) required based in the inputs given by the user, to each particle it gives a random co-ordinate in the space of the frame ensuring that its personal frame of exploration is not going outside the frame given for the algorithm to process we can do this by limiting the space in which the particle can lie. Once all the particles are created and given their coordinates, we append all of them into a list for referencing. We create 2 variables which store the global best results (you'll get to know in further reading) and the co-ordinates of the global best frame which is considered as the target and are common for all particles, 2 more variables one of which stores the state if the particle is found or not(Boolean, initialised with false) and the other stores the number of trails(number initialised with 0) which resulted in an obscure target. From now we use these particles to look into the frames of given footage, for every particle, its coordinate is used to determine its frame. If it is the first processing iteration over the frame, then the personal best results value of the particle is set to 0 and so is global best results value, if target found value is false and personal best co-ordinates are randomized to some location. Once its frame is determined the present frame and the target photo is sent to a function (the other function in this process), in this function Orb classifier is used to generate the key points in the both the frames and their descriptors are generated at the same time together known as features, then we find the matches between the features using a machine learning algorithm (KNN) and the no of good matches (good matches are determined based the difference between the matched matches) are returned, and lets call this returned value as fitness Value, this value is then compared with its present personal best values(along with the personal best coordinates) ,or some thresholds, then the personal best is updated, along with target found is set to true and target lost is set to 0 if it is more (more no of matches), and then the fitness value is compared with the global best value , or some threshold, and updates the global best value and the coordinates, along with gc is set to its original value if the thresholds satisfy, else and target lost value is incremented, gc is set to 0, and is target lost value is greater than threshold, the target found value is set to false. Once this is done , then we determine the direction and jump magnitude of the particle for the next iteration, this movement along x and y direction comes from the sum of 3 components - previous direction, movement towards personal best and movement towards the global best.

Here, 'wt' is the magnitude at which the particle will tend to move in its previous direction and magnitude, 'pc' is the magnitude at which the particle will tend to move towards the personal best co-ordinates, 'gc' is the magnitude at which the particle will tend to move towards the global best co-ordinates, 'R' is the random number between 0 and 1 , we need it to have a randomness in the magnitude of the movements so the they don't settle at a place faster and we have large and favourable space for more exploration, once Vx and Vy are determined we add these values to the co-ordinates of the particle and ensure that it is not going out of the frame with some conditions, this is all what happens in an algorithm iteration of a particle, the no of iterations to be performed on each frame is determined by the user, the co-ordinates of each particle are remembered in video frame so there more chance that the particle is near its target in the next video frame, and make's it easier to track the target. Once all the iterations for all the particles is done the global best co-ordinates are used to draw a green frame around their location, which says it is best result they found in that frame and this output frame is sent to Output process through the AO Pipe.

The Output Process waits for the frame at the other end of AO Pipe once it gets a frame it then prints the frame and comes back to wait for the next frame, this is all what the Output Process does.

As the user we'll only see the output printed by the Output Process but Algorithm process will always process the next frame, and the Parent Process is reading the future frame (the one after the next frame) at the same time, hence makes it possible to execute at an optimum speed.

Algorithm :

Get no of particles

Get no of iterations

Get the video footage and target image

Read a frame

Determine the space where the particles can lie

Set the global best location as some location

For every particle:

Set a random location in the frame as its co-ordinates

Set random values as its velocity

Set random location as personal best location

Set target found to be false

Set target Lost to be 0

For every frame:

Set global best value to be 0

If target found is false:

For every particle:

Set random velocity

Set a random personal best location

For every iteration:

For every particle:

Obtain the frame of particle

Check similarities between particle frame and target frame, say them fitness value

If first iteration:

Set particle best value to 0

If particle personal best value < fitness value:

Set particle personal best value to fitness value

Set particle personal best location to present location

If particle personal best value > global best value and > some threshold:

Set global best value to particle personal best

Set global best location to particle personal best location

Set target found value to true

Set target lost value to 0

Set global weight to given value

Else:

Increment target lost value

If target lost value is greater than some threshold:

Set global weight to 0

Set target found value to false

Determine the velocity , the sum of product of [personal weight, random value and previous velocity], [personal best weight, random value, and displacement from present location to personal best location], [global weight, random value, and displacement from present location to global best location]

Move the present location by adding present location and velocity

Ensure present location is in the frame

Draw the required shapes on the frame based on the particle and global locations and their values

Print the frame