# How the program works:

## Overview

The program would first try to get the background image by averaging every frame in the program, then for each of the frame, it would do a diff with the background image, use the diffed image as a mask, and further separate the background & foreground image.

## Average the images to get the Background

When it’s averaging every frame, it uses the function below to get the alpha value (`fr` is the frame number):

It then merge the frame with the result frame, according to function:

Note that then doing average, one can also set alpha to be:

And the background image would then be:

But the first method (used in the code) can be also used real-time, i.e., when we only have a video stream and total frame number is unknown.

## Background-Removal

For each of the image in the video, it would do a background-removal with the calculated background-frame above.

It would first calculate the difference of the grayscale of the background and the current frame, do a diff and further apply a threshold:

**thresh,biImg = cv2.threshold(diffImg, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)**

to get binary-image, where the non-zero values are the places that has foreground.

It then do a dilate:

**fg = cv2.dilate(biImg, None, iterations = 2)**

to avoid the edge of the foreground being removed.

It then use bitwise\_and to apply the **fg** as a mask:

**res = cv2.bitwise\_and(img, img, fg, fg)**

and the res is the foreground image after the background removal.

To better illustrate, it erode the biImg and apply the inverse-threshold:

**bgtemp = cv2.erode(biImg, None, iterations = 3)**

**thresh2, bg = cv2.threshold(bgtemp, 0, 255, cv2.THRESH\_BINARY\_INV)**

to show the original-background.

Note that the erode here is to correspond to the dilute applied to the fg mask, therefore removes the unnecessary edges in the background mask.