**Algorithm for the YouTube thumbnail generator**

**Overview and reasoning for the project:**

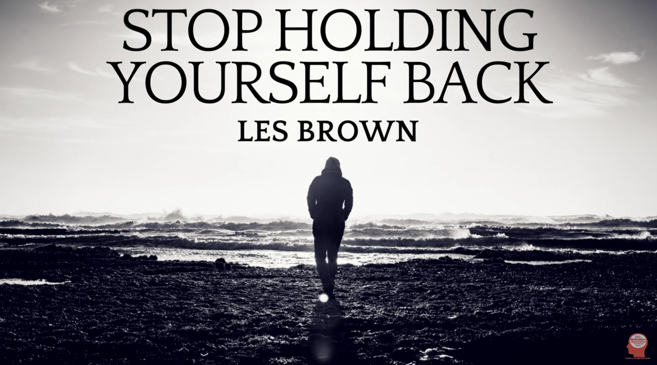
Creating clear, concise and compelling thumbnails that attract clicks on YouTube can be a lengthy and laborious task. Even if you are working quickly and efficiently, it is unlikely that you will get more than 10 done in an hour (the number is usually more like 6-8 per hour).

It is not uncommon for people to bulk upload scheduled content across multiple channels, especially if the content can be generated quickly. Say if you were to schedule to upload 1 video per day for a year, each to 5 channels, that is 1825 videos that is going to take a minimum of 182.5 hours to create the thumbnails.

Further to this, if you are generating content quickly, then a large amount of your time spent on your YouTube channel will be in creating the thumbnails. This is not only suboptimal but also extremely boring.

Take a look at the images below:

A close up of a dog

Description automatically generated

Both these videos did extremely well, each receiving over 500k views each. It is undoubtable that the thumbnails have played a part in the success of these videos. Notice, however, that the placement of the text is strategic, and blends in nicely with the background image.

Whilst not impossible, these thumbnails would be very difficult to generate autonomously. Take a look at the images below:

These videos still did very well, and the structure of the text relative to the image is generic and hence much easier to generate autonomously, creating thumbnails like this is the focus of this project.

**Project description:**

We are going to be inputting 2 folders into this algorithm, each containing an equal number of files.

The first folder will contain a list of background images (with dimensions 1280px x 720px) and the second folder will contain a list of .mp3 files.

Each .mp3 file will be in the form:

LARGE TEXT – SMALL TEXT.mp3.

To generalize this form, let’s label everything up to one space before the dash as component 1 (i.e. “LARGE TEXT” in this case) and label everything from one space after the dash up till the .mp3 as component 2 (i.e. “SMALL TEXT in this case).

So just to clarify, in the 2 examples at the end of the last page, the two mp3 files to create those thumbnails would be:

* LOVE YOURSELF – LES BROWN.mp3

Component 1 = LOVE YOURSELF

Component 2 = LES BROWN

* BEST REVENGE IS MASSIVE SUCCESS – LES BROWN.mp3

Component 1 = BEST REVENGE IS MASSIVE SUCCESS

Component 2 = LES BROWN

We are concerned with projecting component 1 and component 2 onto the image in a certain format that will be discussed shortly.

In these examples, the videos for which the thumbnails are made for are motivational videos, and component 1 is the name/topic of the video and component 2 is the motivational speaker.

I will now outline a 2-step process on how to create one thumbnail (with a single image file and mp3 file), if this can be done then it should be fairly easy to create a for loop and do it many times over.

1. Take the mp3 file in the form “LARGE TEXT – SMALL TEXT.mp3” and project component 2 onto the image as shown below:

A picture containing sky, ceiling, light, outdoor

Description automatically generated

Now define:

* F2 = font size of component 2 (SMALL TEXT).
* DFB = (Distance from bottom) The distance from the bottom of component 2 to the bottom of the image.
* l1 = Horizontal distance from the left-hand side of component 2 (SMALL TEXT) to the left-hand side of the image.
* l2 = Horizontal distance from the right-hand side of component 2 (SMALL TEXT) to the right-hand side of the image.

(Aside: I’m about to include a set of constants, c1, c2, … c8. Obviously, the code will not run without inputting values for c1, c2, … c8. Please could you return the code to me using c1, c2, … c8 throughout and then define the constants at the start of the code so that I can trial and error with these values to see what looks good without having to go through the entire code to pick out the values and change them manually).

Now, we want to:

Maximize F2  c1

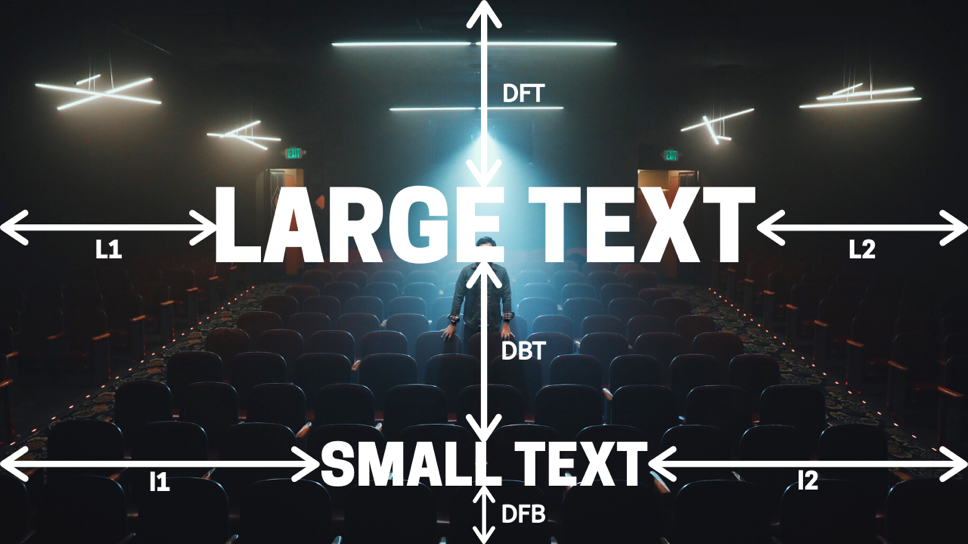
Subject to:

DFB = c2

l1 = l2

l1, l2  c3

1. Take the mp3 file in the form “LARGE TEXT – SMALL TEXT.mp3” and component 1 onto the image as shown below:



Now define:

* F1 = font size of component 1 (LARGE TEXT).
* DBT = (Distance between texts (or components)) The distance from the bottom of component 1 (LARGE TEXT) to the top of component 2 (SMALL TEXT).
* DFT = (Distance from top) The distance from the top of component 1 to the top of the image.
* L1 = The horizontal distance from the left-hand side of component 1 (LARGE TEXT) to the left-hand side of the image.
* L2 = The horizontal distance from the right-hand side of component 1 (LARGE TEXT) to the right-hand side of the image.

Now, we want to:

Maximize F1  c4

Subject to:

DBT = DFT

DBT, DFT  c5

L1 = L2

L1, L2  c6

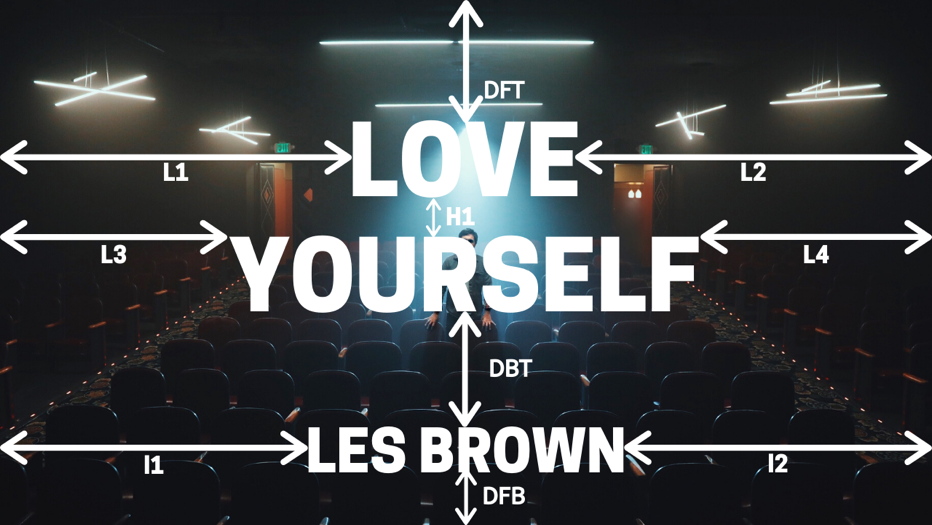
This is assuming that component 1 stays on the first line, which often will not be the case, as was shown in the 2 examples for Les Brown. (Using the same image as above, but with the mp3 file of “LOVE YOURSELF – LES BROWN.mp3”)

To combat this, we are going to need to use an IF statement that breaks component 1 onto 2 lines if L1 and L2 have reached their lower limit, but F1 has not yet reached its upper limit, namely:

IF L1, L2 = c6 AND F1 < c4 {

Break component 1 onto 2 lines as shown in the image below}

(This is using the same image as above, but with the mp3 file of “LOVE YOURSELF – LES BROWN.mp3” instead)



Now define:

* F1 = font size of component 1 (LOVE YOURSELF).
* DBT = (Distance between components) The distance from the bottom of component 1 (LOVE YOURSELF) to the top of component 2 (LES BROWN).
* DFT = (Distance from top) The distance from the top of component 1 to the top of the image.
* L1 = The horizontal distance from the left-hand side of line 1 of component 1 (LOVE) to the left-hand side of the image.
* L2 = The horizontal distance from the right-hand side of line 1 of component 1 (LOVE) to the right-hand side of the image.
* L3 = The horizontal distance from the left-hand side of line 2 of component 1 (YOURSELF) to the left-hand side of the image.
* L4 = The horizontal distance from the right-hand side of line 2 of component 1 (YOURSELF) to the right-hand side of the image.
* H1 = Distance between the bottom of line 1 of component 1 and the top of line 2 of component 1.

Now, we want to:

Maximize F1  c4

Subject to:

DBT = DFT

DBT, DFT  c5

L1 = L2

L1, L2  c6

L3 = L4

L3, L4  c6

H1  c7 AND H1  c8

Finally, we want to allow component 1 to pass onto a maximum of 3 lines if the upper limit for its font size – F1, has not yet being reached. Again, we need to use another IF statement for this:

IF (L1, L2 = c6 OR L3, L4 = c6) AND F1 < c4 {

Break component 1 onto 3 lines as shown in the image below}

(This is using the same image as above, but with the mp3 file of “BEST REVENGE IS MASSIVE SUCCESS – LES BROWN.mp3” instead)

A picture containing wall, black, indoor

Description automatically generated

Now define:

* F1 = font size of component 1 (BEST REVENGE IS MASSIVE SUCCESS).
* DBT = (Distance between components) The distance from the bottom of component 1 (BEST REVENGE IS MASSIVE SUCCESS) to the top of component 2 (LES BROWN).
* DFT = (Distance from top) The distance from the top of component 1 to the top of the image.
* L1 = The horizontal distance from the left-hand side of line 1 of component 1 (BEST REVENGE) to the left-hand side of the image.
* L2 = The horizontal distance from the right-hand side of line 1 of component 1 (BEST REVENGE) to the right-hand side of the image.
* L3 = The horizontal distance from the left-hand side of line 2 of component 1 (IS MASSIVE) to the left-hand side of the image.
* L4 = The horizontal distance from the right-hand side of line 2 of component 1 (IS MASSIVE) to the right-hand side of the image.
* L5 = The horizontal distance from the left-hand side of line 2 of component 1 (SUCCESS) to the left-hand side of the image.
* L6 = The horizontal distance from the right-hand side of line 2 of component 1 (SUCCESS) to the right-hand side of the image.
* H1 = Distance between the bottom of line 1 of component 1 and the top of line 2 of component 1 AND distance between the bottom of line 2 of component 1 and the top of line 3 of component 1.

Now, we want to:

Maximize F1  c4

Subject to:

DBT = DFT

DBT, DFT  c5

L1 = L2

L1, L2  c6

L3 = L4

L3, L4  c6

L5 = L6

L5, L6  c6

H1  c7 AND H1  c8

Output

I am looking for a program which will carry out this task and create as many thumbnails as I want. In other words, I would like to be able to input 2 folders into the program, one containing images (in png or jpg) and one containing a list of mp3 files in which the name of the file is to be extracted in two components (separated by the dash), along with 8 values for the constants c1, c2,…, c8 and receive an output of a single folder containing image files (png or jpg) with the compiled thumbnails. It would also be nice if I could change up the font, color and style of the text to be placed on the images. Thank you and enjoy!