Some thoughts on NormalizedGridDistortion

Bernhard Liebl

October 2020

Code discussed here: `https://github.com/albumentations-team/albumentations/blob/538f67d0b062905e114610b7af86d2b7f62bf902/albumentations/augmentations/functional.py#L1114`.

For simplicity's sake, we only look at the vertical `y` case, the horizontal `x` is analogous.

The coordinates we look at end up in OpenCV's REMAP function. Their meaning is as follows:

`start` and `end` are the coordinates in the target image (the new augmented image to which we write), whereas `cur` is a source coordinate in the image from which we read. It advances at a different speed than the target.

To normalize: when `end` reaches `height`, i.e. when we hit that block which hits the border of the target image we want `cur` to reach `height` as well.

When does this happen though? It depends on `height` and `num_steps` (see Tables 1 and 2):

- If $\frac{height}{num\_steps}$ is an integer, this happens when `idx` = `num_steps - 1`. The last iteration is a no-op.
- If $\frac{height}{num\_steps}$ is not an integer, this happens when `idx` = `num_steps`.

Note that in both cases, the last step in not a full Y_STEP, but smaller (even zero). If we later normalize contributions across these segments, we need to take this into account: the tiny bit at the end should not contribute the same amount of source image as the inner parts (i.e. this last part should do a linearly smaller step in the source image as well). If we don't compensate for this, we will get an unwanted distortion due to the way the remapping gets set up.

For the moment, and for simplicity though, first assume that all steps are the same size in the target image. To assure that no content is distorted outside the frame we want to make sure that `cur` stays ≤ `height` at all times.

To understand how to achieve this, let us look at the value of `cur`. At the end of an iteration with some `idx`, the value of `cur` for that iteration, i.e. cur_{idx}, is

Table 1: `idx` and `cur` for `height` =256 and `num_steps`=8.

idx	cur
0	32
1	64
2	96
3	128
4	160
5	192
6	224
7	256
8	256

Table 2: `idx` and `cur` for `height` =256 and `num_steps`=10.

idx	cur
0	25
1	50
2	75
3	100
4	125
5	150
6	175
7	200
8	225
9	250
10	256

$$cur_{idx} = y\_step \sum_{i=0}^{idx} ysteps[i] \tag{1}$$

i.e.

$$cur_{idx} = \lfloor \frac{height}{num\_steps} \rfloor \sum_{i=0}^{idx} ysteps[i] \tag{2}$$

Given the assumption from above, that we want $cur_{num\_steps} = height$, we obtain the following condition for normalization:

$$\frac{height}{\lfloor \frac{height}{num\_steps} \rfloor} = \sum_{i=0}^{num\_steps} ysteps[i] \tag{3}$$

In other words, we need to normalize `y_steps`, such that the sum the of all `y_steps` elements matches the value on the left.

To compensate for the smaller last step size mentioned earlier, we can now simply adjust the last `ysteps` elements accordingly, before normalizing, such that its contribution matches that of the matching source image step.