Operating System Lab

Lab 5

Kunal Kumar - 170010012
Karthik VV - 170010016

---

1. Image Transformations Used:

1. **RGB to GrayScale** : We used a weighted average method to grayscale. Since red has more wavelengths of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.

It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.

So the new equation that form is:

New grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B)).

According to this equation, Red has contributed 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

2. **Sobel Edge Detection:** The sobel Operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

The operator consists of a pair of 3×3 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°. This is very similar to the Roberts Cross operator. These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

2. Proof of Correctness:

The operations applied are,

T1 o $[A]_{3x3}$

T2 o $[B]_{3x3}$

Here $[A]_{3x3}$ and $[B]_{3x3}$ during time of operation should never have common elements, otherwise partially modified values by Tj will be picked up by Ti which will result in wrong output.

To ensure this doesn't happen we have kept a size 2 array [T1ed_till[2]] available to both T1 and T2 functions as:-

Parts:

2.1.a: global -- since global variables are accessible by threads

2.1.b: global -- since global variables are accessible by threads

2.2: it is pushed to a seperate shared_memory of size(int) *2 other than the pix_map

2.3: it is pushed to separate pipe other than the pix_map

3. Observations:

Part	Method	Time
part1	Sequential	16810
part2_1a	Threading + atomic operations	28086
part2_1b	Threading + semaphores	62018
part2_2	Diff_processes + shared_memory + semaphores	53527
part2_3	Diff_processes + Piping	66529

Conclusion: From the above table we can figure out that introducing parallelism whether via threading or multiprocessing did not help in decreasing the time consumed. This points to the fact that the overhead of process synchronisation and ensuring correctness methods are having a greater incremental effect on the run_time than is decreased by the parallelism; i.e the critical section is smaller than the non-critical section, So sequentialization the critical section is far too expensive to decrease run time.