Vietnam National University - Ho Chi Minh City
UNIVERSITY OF SCIENCE

Image and Video Processing

Individual Lab 01

Student: Tran Kien Quoc - 19127535

Contents

1 Program

1.1 Compilation

The program can be compiled using Qt Creator or qmake build system with the G++ compiler.

```
qmake
make
```

The compiler produces an executable `DIP_Lab01`.

1.2 Usage

Command line:

```
./DIP_Lab01 <arguments>
```

- `-rgb2gray <input> <output>`: Convert `<input>` into grayscale, written into `<output>`.
- `-rgb2hsv <input> <output>`: Convert `<input>` into HSV (color quantization), with 17 colors (Hue), 17 levels of saturation and 256 levels of intensity, written into `<output>`.
- `-drawhist <input> <output>`: Draw histogram.
- `-equalhist <input> <output>`: Draw histogram.
- `-bright <b> <input> <output>`: Change brightness by *b*.
- `-contrast <c> <input> <output>`: Change contrast by *c*.

2 Solutions

2.1 Load the input image

Use `cv::imread` with a `cv::Mat`. For example:

```
image = cv::imread(fileinp, cv::IMREAD_COLOR);
```

2.2 Save an output image

Use `cv::imwrite` with `fileout` is the output file name stored as a string and `outImage` is a `cv:::Mat`

```
cv::imwrite(fileout, outImage);
```

2.3 RGB to grayscale

Iterate through the CV Matrix with a pointer for each row. For each pixel with color (R, G, B), the grayscale value is calculated as: $Gray = (3R + 6G + B)/10$. Then each value of RGB is assigned to the $Gray$ value.

2.4 RGB to HSV

Algorithm to convert RGB to HSV:

- $M = max(R, G, B), m = min(R, G, B)$
- $V \leftarrow max(R, G, B)$
- If $M = m$, return $(H, S, V) = (0, 0, V)$
- $S \leftarrow 1.0 * D/M$
- If $M == R$, $H \leftarrow (60 * (G - B)/D + 360)\%360$
- If $M == G$, $H \leftarrow (60 * (B - R)/D + 360)\%360$
- If $M == B$, $H \leftarrow (60 * (G - R)/D + 360)\%360$
- $H \leftarrow H/360$
- return H, S, V

For each pixel, apply the algorithm.

2.5 Change brightness

For each pixel with R, G, B value:

$$R \leftarrow R + b$$
$$G \leftarrow G + b$$
$$B \leftarrow B + b$$

2.6 Change contrass

For each pixel with R, G, B value:

$$R \leftarrow R * c$$
$$G \leftarrow G * c$$
$$B \leftarrow B * c$$

2.7 Draw histogram

Convert each pixel into HSV color space with $nH = 17, nS = 17, nV = 17 (H \in [0, 16], S \in [0, 16], V \in [0, 16])$. The histogram size is 4913 x 1000.

`CalcHistogram` generates `histMatrix`, 3D `cv::Mat` to store the frequency of each color. Normalize the frequencies into the $[0, 999]$ range.

2.8 Equalize histogram

Equalize the intensity histogram. For each pixel with R, G, B values, convert into H, S, V colors space with $V \in [0, 255]$. Use an array T to count the frequency of the intensity V values.

Calculate cumulative sum of T.

For each element T_i in T:

$T_i \leftarrow \frac{255}{M*N} * T_i$, with $M * N$ is the image size.

Then, for each pixel with intensity V_i, change V_i into $T[V_i]$, and convert back into RGB.