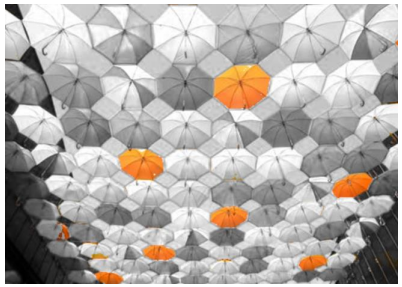
Color Processing

Sung Soo Hwang

Color Processing

- Usage of HSI
  - Intensity images is decoupled
    - Can change the intensity of the image only
  - Color Slicing
    - Find the pixels in the range of the desired color in the **Hue**-channel
    - Set all the other pixels to 0 in the **Saturation**-channel
  - Color Conversion
    - By accessing the **Hue**-channel we can change the regions of colors

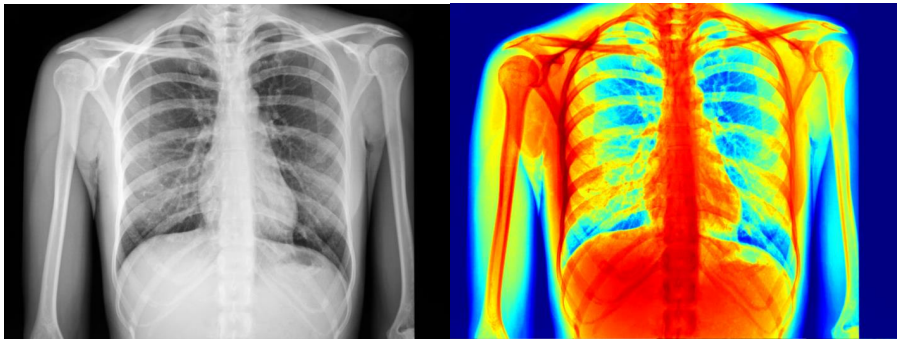





https://www.it.uu.se/edu/course/homepage/bild1/vt09/lectures/L06VT09.pdf

Color Processing

- Pseudo Coloring
  - The eye can distinguish between only about 30-50 different shades of gray.
  - But can distinguish about 100k ~ 10m colors.
  - Useful to display gray scale images using color to visualize the information better
  - Important to include a color scale in the images to understand what the colors illustrate
  - Example of pseudo-coloring :
    - Intensity Slicing
      - Each intensity is assigned a color



https://www.it.uu.se/edu/course/homepage/bild1/vt09/lectures/L06VT09.pdf

Color Balancing

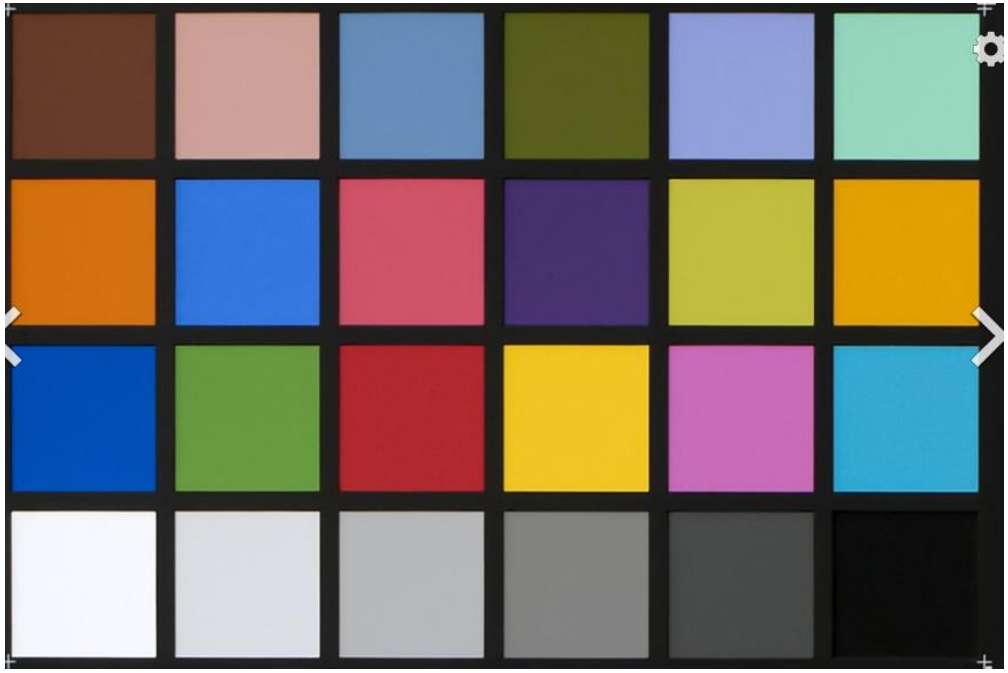
- Definition
  - Global adjustment of the intensities of the colors

- Simple way of color balancing
  - Scale R,G,B components so that objects which are believed to be neutral appear so

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R'_w & 0 & 0 \\ 0 & 255/G'_w & 0 \\ 0 & 0 & 255/B'_w \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

Color Balancing

- Using color checker



- Estimate white color in an image
  - Gray world assumption
    - In a normal well color balanced photo, the average of all the colors is a neutral gray
  - White pixels are also brightest

Color Processing - Usage of HSI

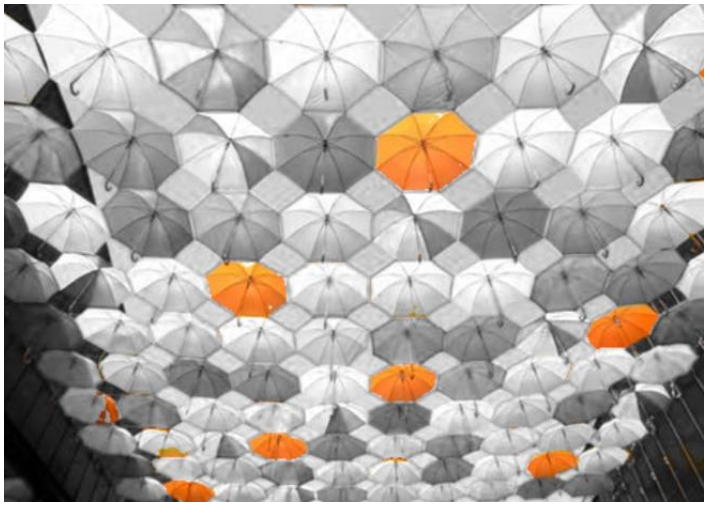
- Example code:

```
int main() {
	Mat image = imread("colorful.jpg");
	Mat HSV, intensity_change, mask_out, change_color;
	vector<Mat> ic(3);
	vector<Mat> mo(3);
	vector<Mat> cc(3);

	int rows = image.rows;
	int cols = image.cols;
	uchar* h;
	uchar* s;
	uchar* v;

	cvtColor(image, HSV, CV_BGR2HSV);
	split(HSV, ic);
	split(HSV, mo);
	split(HSV, cc);


	//eqaulizing the histogram of I mat
	equalizeHist(ic[2], ic[2]);

	//masking out except orange
	for (int i = 0; i < rows; i++) {
		h = mo[0].ptr<uchar>(i);
		s = mo[1].ptr<uchar>(i);
		for (int j = 0; j < cols; j++) {
			if (h[j] > 9 && h[j] < 23) s[j] = s[j];
			else s[j] = 0;
		}
	}
```

```
	//changing all colors
	for (int i = 0; i < rows; i++) {
		h = cc[0].ptr<uchar>(i);
		s = cc[1].ptr<uchar>(i);
		for (int j = 0; j < cols; j++) {
		if (h[j] + 50 > 255) h[j] = h[j] + 50 - 255;
		else h[j] += 50;
		}
	}

	merge(ic, intensity_change);
	merge(mo, mask_out);
	merge(cc, change_color);
	cvtColor(intensity_change, intensity_change, CV_HSV2BGR);
	cvtColor(mask_out, mask_out, CV_HSV2BGR);
	cvtColor(change_color, change_color, CV_HSV2BGR);

	imshow("image", image);
	imshow("intensity change", intensity_change);
	imshow("mask out", mask_out);
	imshow("change color", change_color);

	waitKey(0);
	return 0;
}
```

Color Processing

- Usage of HSI - results

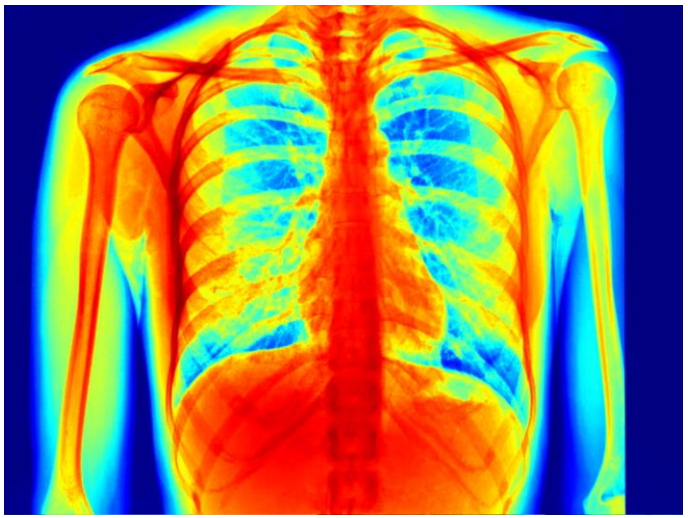


Color Processing

- Pseudo coloring

```
int main() {
	Mat gray = imread("xray.jpg", 0);
	Mat color;
	// Applies a colormap on a given image
	// gray: src, color: dst, COLORMAP_JET: the color map to apply
	applyColorMap(gray, color, COLORMAP_JET);
	imshow("gray", gray);
	imshow("image", color);
	waitKey(0);
	return 0;
}
```

Class	Scale
COLORMAP_AUTUMN	
COLORMAP_BONE	
COLORMAP_COOL	
COLORMAP_HOT	
COLORMAP_HSV	
COLORMAP_JET	
COLORMAP_OCEAN	
COLORMAP_PINK	
COLORMAP_RAINBOW	
COLORMAP_SPRING	
COLORMAP_SUMMER	
COLORMAP_WINTER	



https://www.verywellhealth.com/chest-x-rays-for-lung-cancer-diagnosis-4107046
https://docs.opencv.org/2.4/modules/contrib/doc/facerec/colormaps.html?highlight=applycolormap

Color Balancing

- Example code

```
int main() {
	Mat image = imread("lena.png");
	Mat result;
	vector<Mat> ch(3);
	int b_sum = 0, g_sum = 0, r_sum = 0;
	int b_avg, g_avg, r_avg, b_tmp, g_tmp, r_tmp;
	if (image.empty()) {
		cerr << "read fail" << endl;
		exit(-1);
	}
	int rows = image.rows;
	int cols = image.cols;
	int pixno = rows * cols;

	// split by B, G, R channel
	split(image, ch);
	uchar* b;
	uchar* g;
	uchar* r;
	// calculate each channel's average
	for (int i = 0; i < rows; i++) {
		b = ch[0].ptr<uchar>(i);
		g = ch[1].ptr<uchar>(i);
		r = ch[2].ptr<uchar>(i);
		for (int j = 0; j < cols; j++) {
			b_sum += b[j];
			g_sum += g[j];
			r_sum += r[j];
		}
	}
```

```
	b_avg = b_sum / pixno;
	g_avg = g_sum / pixno;
	r_avg = r_sum / pixno;

	// color balancing using gray world assumsption
	for (int i = 0; i < rows; i++) {
		b = ch[0].ptr<uchar>(i);
		g = ch[1].ptr<uchar>(i);
		r = ch[2].ptr<uchar>(i);
		for (int j = 0; j < cols; j++) {
			// to prevent overflow
			b_tmp = (128 * b[j]) / b_avg;
			if (b_tmp > 255) {
				b[j] = 255;
			}
			else {
				b[j] = b_tmp;
			}

			g_tmp = (128 * g[j]) / g_avg;
			if (g_tmp > 255) {
				g[j] = 255;
			}
			else {
				g[j] = g_tmp;
			}
```

Color Balancing

- Example code

```
                        r_tmp = (128 * r[j]) / r_avg;
                        if (r_tmp > 255) {
                                r[j] = 255;
                        }
                        else {
                                r[j] = r_tmp;
                        }
                }
        }

        // merge 3 channel's image
        merge(ch, result);

        imshow("image", image);
        imshow("result", result);
        waitKey(0);
        return 0;
}
```