# Color Conversion, Split/Merge

Sung Soo Hwang

# Color conversion

- Color space conversion
  - void cvtColor(Mat src, Mat dst, int code, int dstCn =0)
    - Convert an image frame one color space to another
    - Code: CV_BGR2GRAY, CV_BGR2HSV, CV_BGR2YCrCb, CV_BGR2Lab, ……)
    - dstcn: destination channel number. If 0, automatically determined by src and dst

# Color conversion

- Color space conversion
  - Example code

```
int main() {
	Mat image, image_YUV;

	image = imread("lena.png");

	cvtColor(image, image_YUV, CV_BGR2YUV);
}
```

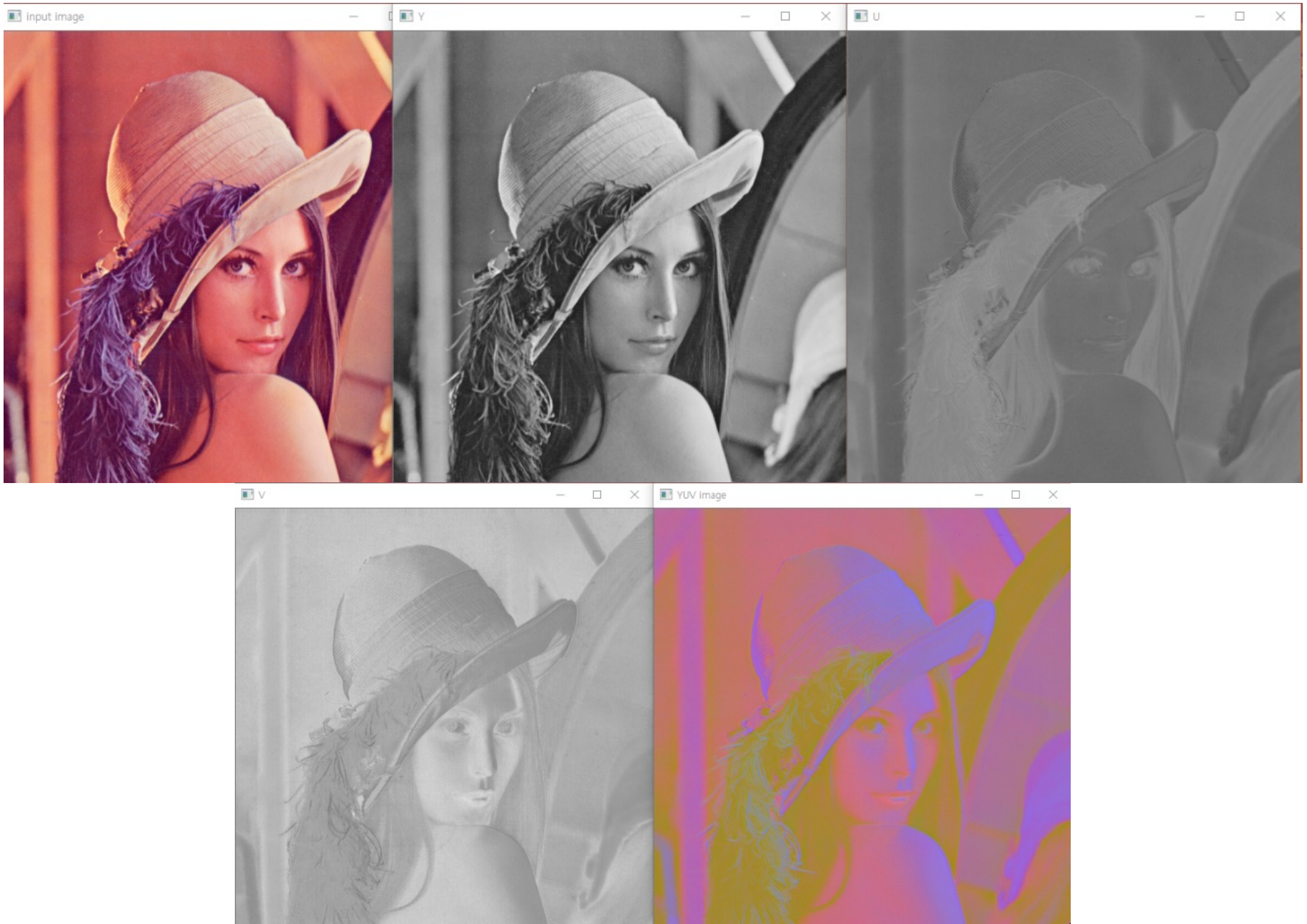
# Split/Merge

- Split/Merge
  - void split(Mat src, Mat* mv)
    - Splits multi-channel array into separate single-channel arrays
    - mv: output array (vector of arrays) mv[c][I] = src[I]* the number of arrays must match src.channels()
  - merge(InputArrayOfArray mv, OutputArray dst): reverse of split
    - mv: vector of matrices all of the matrices in mv must have same size and depth
    - dst : output array of the same size and depth as mv[0]

# Mat Operator

- Color space conversion
  - Example code

```
int main() {
	Mat image, image_YUV, dst;
	Mat yuv_channels[3];

	image = imread("lena.png");

	cvtColor(image, image_YUV,
	COLOR_BGR2YUV);

	split(image_YUV, yuv_channels);

	merge(yuv_channels,3,dst);

	imshow("input image", image);
	imshow("Y", yuv_channels[0]);
	imshow("U", yuv_channels[1]);
	imshow("V", yuv_channels[2]);
	imshow("YUV image", dst);

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


input image
Y
U
V
YUV image

# Color Processing

Sung Soo Hwang

# 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, COLOR_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 j = 0; j < rows; j++) {
		h = mo[0].ptr<uchar>(j);
		s = mo[1].ptr<uchar>(j);
		for (int i = 0; i < cols; i++) {
			if (h[i] > 9 && h[i] < 23) s[i] = s[i];
			else s[i] = 0;
		}
	}
	//changing all colors
	for (int j = 0; j < rows; j++) {
		h = cc[0].ptr<uchar>(j);
		s = cc[1].ptr<uchar>(j);
		for (int i = 0; i < cols; i++) {
		if (h[i] + 50 > 179) h[i] = h[i] + 50 - 179;
		else h[i] += 50;
		}
	}

	merge(ic, intensity_change);
	merge(mo, mask_out);
	merge(cc, change_color);
	cvtColor(intensity_change, intensity_change, COLOR_HSV2B
	GR);
	cvtColor(mask_out, mask_out, COLOR_HSV2BGR);
	cvtColor(change_color, change_color, COLOR_HSV2BGR);

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

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

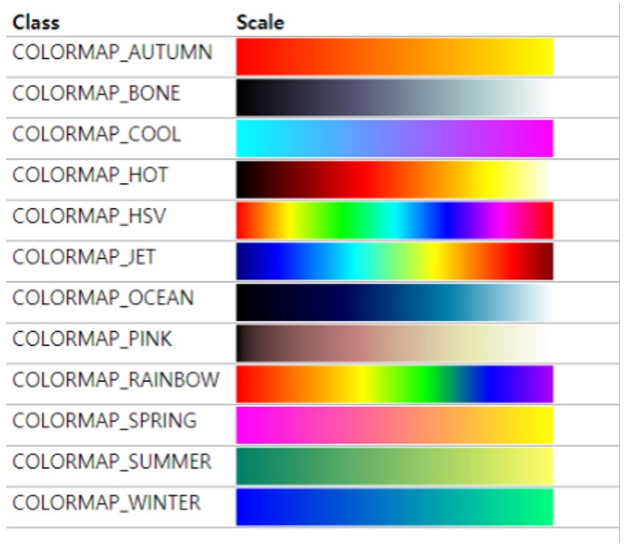
# Color Processing

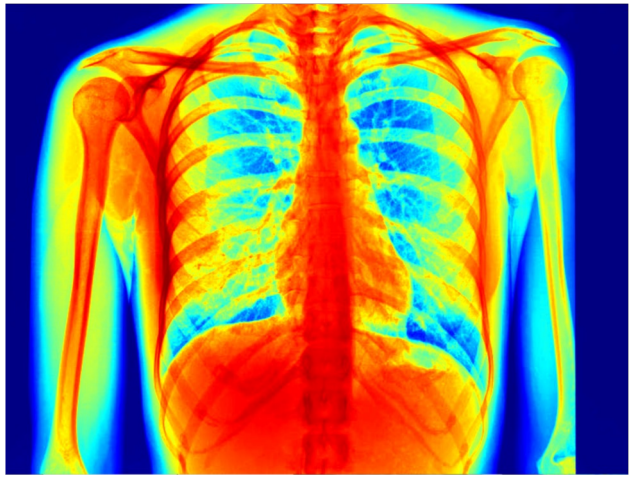
- Usage of HSI - results

# 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;
}
```



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 Processing

- White balancing

```
void white_balacing(Mat img) {
   Mat bgr_channels[3];
   split(img, bgr_channels);

   double avg;
   int sum,temp,i, j, c;

   for (c = 0; c < img.channels(); c++) {
      sum = 0;
      avg = 0.0f;
      for (i = 0; i < img.rows; i++) {
         for (j = 0; j < img.cols; j++) {
            sum += bgr_channels[c].at<uchar>(i, j);
         }
      }
      avg = sum / (img.rows * img.cols);
      for (i = 0; i < img.rows; i++) {
         for (j = 0; j < img.cols; j++) {
            temp = (128 / avg) * bgr_channels[c].at<uchar>(i, j);
            if (temp>255) bgr_channels[c].at<uchar>(i, j) = 255;
            else bgr_channels[c].at<uchar>(i, j) = temp;
         }
      }
   }
   merge(bgr_channels, 3, img);
}
```

# Color Processing

- White balancing

```
int main() {
    Mat balancing;
    Mat balancing_result;

    balancing = imread("balancing.jpg");
    balancing_result = balancing.clone();
    white_balacing(balancing_result);
    imshow("image", balancing);
    imshow("balancing", balancing_result);
    waitKey(0);
}
```