January 2017
School of Engineering and Technology
Name: Teera Laiteerapong, 118950

Converting Colors from HSI to BGR

```
#include <algorithm>
#include "opencv2/opencv.hpp"

cv::Mat convertHSItoBGR(cv::Mat inputMatImage) {
    const double pi = 3.14159265358979323846;
    double theta = 0.0;
    // fH is hue, fS is saturation, fI is Intensity,
    // fB is blue, fG is green, and fR is red
    double fH = 0.0; double fS = 0.0; double fI = 0.0;
    double fB = 0.0; double fG = 0.0; double fR = 0.0;
    // Create vector that has array of array of size 3
    // hsiPixel is for the inputMatImage
    // bgrPixel is to store each calculated pixel into array
    cv::Vec3b bgrPixel, hsiPixel;
    // matBGRImage is for the final image output
    cv::Mat matBGRImage(
        inputMatImage.rows,
        inputMatImage.cols,
        inputMatImage.type());

    // Loop through each pixel to convert from HSI to BGR
    for ( int i=0; i < inputMatImage.rows; i++ ) {
        for ( int j=0; j < inputMatImage.cols; j++ ) {
            hsiPixel = inputMatImage.at<cv::Vec3b>(i, j);

            // Normalized HSI pixel values
            fH = ceil(static_cast<double>(hsiPixel[0]) * 360.0 / 255.0);
            fS = static_cast<double>(hsiPixel[1]) / 255.0;
            fI = static_cast<double>(hsiPixel[2]) / 255.0;

            // Argument in cos() need to convert from degree to radian
            // The degree in fH (hue) can divided into 3 sections
            // from 0 - 120 degree (red),
            // 120 - 240 degree (blue),
            // and 240 - 360 (green)
            if (fH == 0) {
                fB = fI - fI * fS;
                fG = fI - fI * fS;
                fR = fI + fI * fS * 2.0;
            } else if (fH == 120) {
                fB = fI - fI * fS;
                fG = fI + fI * fS * 2.0;
                fR = fI - fI * fS;
            } else if (fH == 240) {
                fB = fI + fI * fS * 2.0;
                fG = fI - fI * fS;
                fR = fI - fI * fS;
            } else if (0 < fH && fH < 120) {
                if ((60-fH) < 0) {
                    theta = (360 - (60 - fH)) * pi / 180;
                } else {
                    theta = (60 - fH) * pi / 180;
                }
```

```
                fB = fI - fI * fS;
                fG = fI + fI * fS * (1.0 - cos(fH * pi / 180) / cos(theta));
                fR = fI + fI * fS * cos(fH * pi / 180)/cos(theta);
            } else if (120 < fH  && fH < 240) {
                if ((180 - fH) < 0) {
                    theta = (360 - (180 - fH)) * pi / 180;
                } else {
                    theta = (180 - fH) * pi / 180;
                }
                fB = fI + fI * fS * (1 - cos((fH - 120) * pi / 180)/cos(theta));
                fG = fI + fI * fS * cos((fH - 120) * pi / 180) / cos(theta);
                fR = fI - fI * fS;
            } else if (240 < fH && fH < 360) {
                if ((300 - fH) < 0) {
                    theta = (360 - (300 - fH)) * pi / 180;
                } else {
                    theta = (300 - fH) * pi / 180;
                }
                fB = fI + fI * fS * cos((fH - 240) * pi / 180) / cos(theta);
                fG = fI - fI * fS;
                fR = fI + fI * fS * (1.0 - cos((fH - 240) * pi / 180) / cos(theta));
            }

            // Denormalize
            bgrPixel[0] = ceil(fB * 255.0);
            bgrPixel[1] = ceil(fG * 255.0);
            bgrPixel[2] = ceil(fR * 255.0);

            // Put BGR pixel back to image
            matBGRImage.at<cv::Vec3b>(i, j) = bgrPixel;
        }
    }

    return matBGRImage;
}

cv::Mat convertBGRtoHSI(cv::Mat inputMatImage) {
    const double pi = 3.14159265358979323846;
    double fB = 0.0; double fG = 0.0; double fR = 0.0;
    double fH = 0.0; double fS = 0.0; double fI = 0.0;
    cv::Vec3b bgrPixel, hsiPixel;
    cv::Mat matHSIImage(
        inputMatImage.rows,
        inputMatImage.cols,
        inputMatImage.type());

    for ( int i=0; i < inputMatImage.rows; i++ ) {
        for ( int j=0; j < inputMatImage.cols; j++ ) {
            bgrPixel = inputMatImage.at<cv::Vec3b>(i, j);
            // Normalized pixel value
            fB = static_cast<double>(bgrPixel[0])/255.0;
            fG = static_cast<double>(bgrPixel[1])/255.0;
            fR = static_cast<double>(bgrPixel[2])/255.0;

            // Get theta in radian
```

```
            double theta = acos(
                (0.5*((fR-fG)+(fR-fB)))/sqrt(pow(fR-fG, 2)+((fR-fB)*(fG-fB))));

            // Convert radian from range 0~+-Pi to 0~2Pi
            if ( theta < 0 ) {
                theta =(2.0*pi)-theta;
            }

            // Convert radian (0~2*3.14) to degree (0~360)
            theta = theta*(360.0)/(2.0*pi);

            // H channel
            if ( fB <= fG ) {
                fH = theta;
            } else {
                fH = 360-theta;
            }

            // S channel
            fS = 1.0 - (3.0/(fR+fG+fB))*std::min(std::min(fR, fG), fB);

            // I channel
            fI = (fR+fG+fB)/3.0;

            // Denormalize
            hsiPixel[0] = fH/360.0*255.0;
            hsiPixel[1] = fS*255.0;
            hsiPixel[2] = fI*255.0;

            // Put HSI pixel back to image
            matHSIImage.at<cv::Vec3b>(i, j) = hsiPixel;
        }
    }

    return matHSIImage;
}

int main(int argc, char** argv) {
    cv::Mat sourceImage;
    sourceImage = cv::imread("./statue.png", cv::IMREAD_COLOR);

    cv::Mat HSIImage;
    HSIImage = convertBGRtoHSI(sourceImage);

    cv::Mat BGRImage;
    BGRImage = convertHSItoBGR(HSIImage);

    cv::imwrite("./converted_statue.png", BGRImage);

    return 0;
}
```