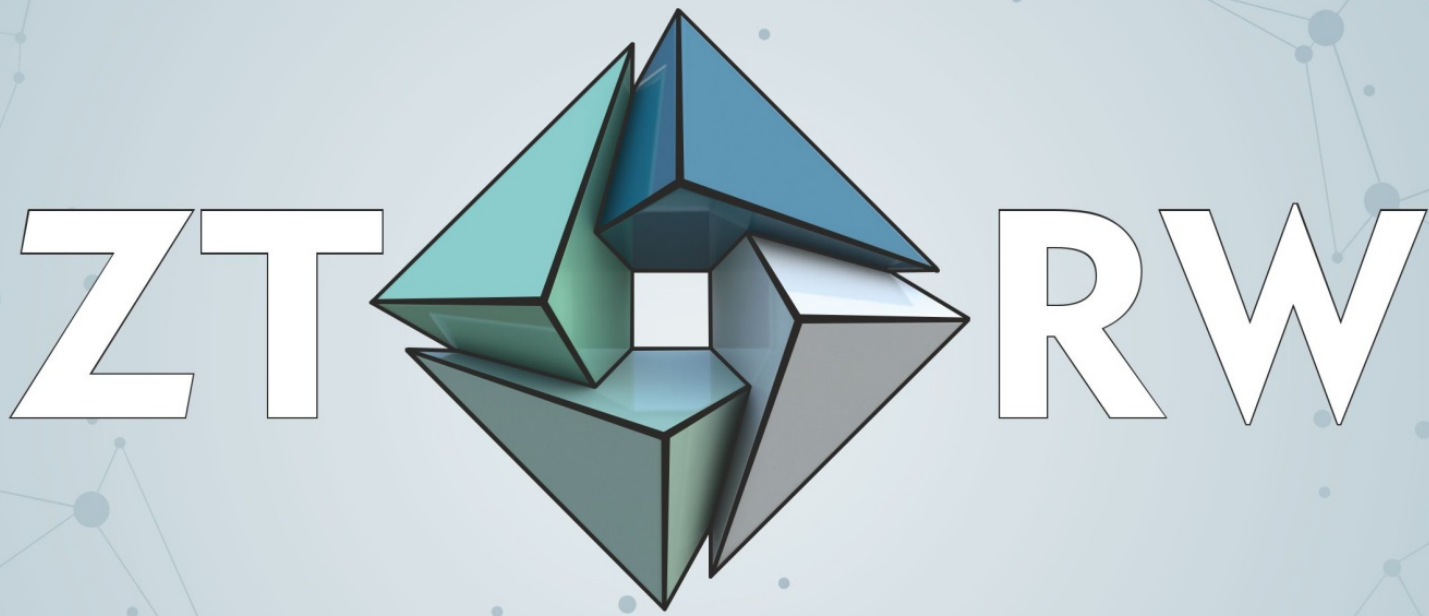
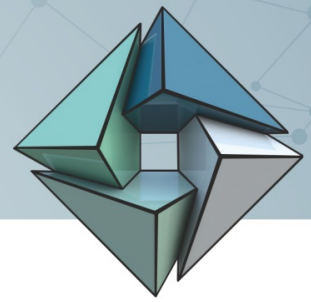

ZT
RW



Digital image processing #3

Color spaces and thresholding

Kamil Szeląg

Warsaw University of Technology
Institute of Micromechanics and Photonics
Virtual Reality Techniques Division

December 13, 2017

Color spaces
RGB,HSV,HSL,XYZ,LAB

- Task #23
  - Load color image
  - Create 4 result images
  - Perform RGB -> HSV conversion
  - Perform RGB -> HSL conversion
  - Perform RGB -> XYZ conversion
  - Perform RGB -> LAB conversion
  - What are color spaces differences
- Task #24
  - Perform Task #23
  - Create array of single channel result images
  - Split every image into channels -> what is visible?
  - Pick one array and perform arithmetic operation on single channel
  - Merge channels - what is changed?

Color spaces
RGB,HSV,HSL,XYZ,LAB

```
//Task #23
        cv::Mat img = cv::imread("lena.bmp");
        cv::Mat hsv_img = cv::Mat(img.size(), img.type());
        cv::Mat hsl_img = cv::Mat(img.size(), img.type());
        cv::Mat xyz_img = cv::Mat(img.size(), img.type());
        cv::Mat lab_img = cv::Mat(img.size(), img.type());

        cv::cvtColor(img, hsv_img, CV_BGR2HSV);
        cv::cvtColor(img, hsl_img, CV_BGR2HLS);
        cv::cvtColor(img, xyz_img, CV_BGR2XYZ);
        cv::cvtColor(img, lab_img, CV_BGR2Lab);

        cv::imshow("Display", img);
        cv::waitKey(0);
        cv::imshow("Display", hsl_img);
        cv::waitKey(0);

        cv::imshow("Display", xyz_img);
        cv::waitKey(0);

        cv::imshow("Display", lab_img);
        cv::waitKey(0);
```

Color spaces
RGB,HSV,HSL,XYZ,LAB

```
//Task #24

        cv::Mat img = cv::imread("lena.bmp");
        cv::Mat hsv_img = cv::Mat(img.size(), img.type());
        cv::Mat hsl_img = cv::Mat(img.size(), img.type());
        cv::Mat xyz_img = cv::Mat(img.size(), img.type());
        cv::Mat lab_img = cv::Mat(img.size(), img.type());

        cv::cvtColor(img, hsv_img, CV_BGR2HSV);
        cv::cvtColor(img, hsl_img, CV_BGR2HLS);
        cv::cvtColor(img, xyz_img, CV_BGR2XYZ);
        cv::cvtColor(img, lab_img, CV_BGR2Lab);

        std::vector<cv::Mat> img_channels;
        cv::split(img, img_channels);
        cv::imshow("First channel", img_channels[0]);
        cv::imshow("Second channel", img_channels[1]);
        cv::imshow("Third channel", img_channels[2]);
        cv::waitKey(0);

        cv::split(hsv_img, img_channels);
        cv::imshow("First channel", img_channels[0]);
        cv::imshow("Second channel", img_channels[1]);
        cv::imshow("Third channel", img_channels[2]);
        cv::waitKey(0);
        // the same for every other image
```

Thresholding
Basic thresholding

- Task #25
  - Load image
  - Convert image to grayscale
  - Try different cv::threshold options. Save result of each operation in separate image

Thresholding

Grayscale thresholding

```
 //Task #25

cv::Mat img = cv::imread("lena.bmp");
cv::Mat grayscale_img;
cv::cvtColor(img, grayscale_img, CV_BGR2GRAY);
cv::Mat img_binary, img_binary_inv, img_otsu;
cv::Mat img_toZero, img_toZero_inv, img_trunc, img_combined;
cv::threshold(grayscale_img, img_binary, 100, 255, CV_THRESH_BINARY);
cv::threshold(grayscale_img, img_binary_inv, 100, 255, CV_THRESH_BINARY_INV);
cv::threshold(grayscale_img, img_otsu, 100, 255, CV_THRESH_OTSU);
cv::threshold(grayscale_img, img_toZero, 100, 255, CV_THRESH_TOZERO);
cv::threshold(grayscale_img, img_toZero_inv, 100, 255, CV_THRESH_TOZERO_INV);
cv::threshold(grayscale_img, img_trunc, 100, 255, CV_THRESH_TRUNC);
cv::threshold(grayscale_img, img_combined, 100, 255, CV_THRESH_OTSU+CV_THRESH_TOZERO_INV);
//there is posibility to merge two or more algorithms

cv::imshow("Display", img_binary);
cv::waitKey(0);
cv::imshow("Display", img_binary_inv);
cv::waitKey(0);
// the same for every other image
```

Thresholding
Color thresholding

- Task #26
  - Load color image with shadows
  - Detect shadow using HSV color space
- Task #27 (optional)
  - Remove shadow from task #26 image

Thresholding

Color thresholding

```
//Task #26
        cv::Mat img = cv::imread("shadow.jpg");
        cv::cvtColor(img, img, CV_BGR2HSV);
        std::vector<cv::Mat> hsvCh;
        cv::split(img, hsvCh);
        cv::normalize(hsvCh[1], hsvCh[1], 0, 255, cv::NORM_MINMAX);
        cv::normalize(hsvCh[2], hsvCh[2], 0, 255, cv::NORM_MINMAX);
        cv::Mat shadow_mask = hsvCh[1] - hsvCh[2];
        cv::threshold(shadow_mask, shadow_mask, 20, 255, CV_THRESH_BINARY);

//Task #27 works only on images with consistent background
        cv::Mat img = cv::imread("shadow.jpg");

        cv::cvtColor(img, img, CV_BGR2HSV);
        std::vector<cv::Mat> hsvCh;
        cv::split(img, hsvCh);
        cv::normalize(hsvCh[1], hsvCh[1], 0, 255, cv::NORM_MINMAX);
        cv::normalize(hsvCh[2], hsvCh[2], 0, 255, cv::NORM_MINMAX);
        cv::Mat sh = hsvCh[1] - hsvCh[2];
        cv::threshold(sh, sh, 20, 255, CV_THRESH_BINARY);
        cv::erode(sh, sh, cv::Mat(), cv::Point(-1, -1), 8);

        cv::Scalar meanSh = cv::mean(img, sh);
        cv::Scalar meanNoSh = cv::mean(img, cv::Scalar::all(255) - sh);
        cv::Scalar diff = -meanSh + meanNoSh;
        cv::Mat res1 = img.clone();

        cv::add(img, diff, res1, sh);
        cv::cvtColor(res1, res1, CV_HSV2BGR);
```

The end

http://ztrw.mchtr.pw.edu.pl