第4回C++輪講

第4回の内容

- 第4回課題

第4回課題

the4thCppLecture_name.hpp，the4thCppLecture_name.cppを作成し以下の関数を名前空間t4cl内に作成せよ
関数run()で全ての関数が実行できるようにすること

- リサイズ関数（バイリニア補完）
  関数名：resize
- 判別分析法
  関数名：discriminantAnalysis
- アフィン変換（引数：上下左右の拡大率，平行移動量，回転角度）
  原点を左上から画像中心へ→拡大縮小→回転→原点を画像中心から左上へ→平行移動
  関数名：affineTransform
  以下の関数を用いて確認せよ

```
void name::t4cl::confirmAffine(const cv::Mat1b & src, const cv::Mat1b &
yourAffine,
                        const double scaleX, const double scaleY,
                        const double transX, const double transY, const double
deg)
{
    cv::Mat1b dst;
    const double rad = (deg * CV_PI) / 180.0;
    cv::Mat1d affineMat = (cv::Mat1d(2, 3) <<
        scaleX * std::cos(rad), -scaleX * std::sin(rad), transX,
        scaleY * std::sin(rad),  scaleY * std::cos(rad), transY);
    cv::warpAffine(src, dst, affineMat, src.size() * 2);            //opencv
affine
    cv::imshow("affineTransformed", yourAffine);
    cv::moveWindow("affineTransformed", 100, 100);
    cv::imshow("affineOpenCV", dst);
    cv::moveWindow("affineOpenCV", 100 + dst.cols, 100);
    cv::waitKey();
    cv::destroyAllWindows();
}
```

- テンプレートマッチング（グレースケール画像）
  関数名：templateMatching
    - 結果表示用の矩形を描く関数も作ること
      関数名：drawRec
- バイラテラルフィルタ
  関数名：bilateral
- 細線化（zhangの方法）
  関数名：thinning

```
// void run(const int num = 0, const std::string & imagePath =
"sample_images/lena.bmp", const std::string & hirakawaPath =
"sample_images/hirakawa.bmp"); //ヘッダではこのように宣言すること
void name::t4cl::run(const int num, const std::string & imagePath, const
std::string & hirakawaPath)
{
    const cv::Mat3b lena(cv::imread(imagePath, cv::IMREAD_COLOR));
    CV_Assert(!lena.empty());
    const cv::Mat1b hirakawa(cv::imread(hirakawaPath, cv::IMREAD_GRAYSCALE));
    CV_Assert(!hirakawa.empty());

    cv::Mat1b gray;
    t3cl::bgr2gray(lena, gray);

    switch (num) {
    default:
    case 1:
        std::cout << "run resize ";
        {
            cv::Mat3b resized;
            // src dst size
            t4cl::resize(lena, resized, cv::Size(640, 320));
            t3cl::imshow("resized", resized);
        }
            std::cout << "\r" << "complete resize" << std::endl;
        if (num != 0) break;
    case 2:
        std::cout << "run discriminantAnalysis";
        {
            cv::Mat1b binary;
            t3cl::binarization(gray, binary, discriminantAnalysis(gray));
            t3cl::imshow("discriminantAnalysis", binary);
        }
            std::cout << "\r" << "complete discriminantAnalysis" << std::endl;
        if (num != 0) break;
    case 3:
        std::cout << "run affineTransform";
        {
            cv::Mat1b affineTransformed;
            constexpr double scaleX(0.8), scaleY(0.8);
            constexpr int transX(100), transY(-50), deg(-40);
            // src dst scaleX yscaleY transX transY degree
            t4cl::affineTransform(gray, affineTransformed, scaleX, scaleY,
```

```
transX, transY, deg);
            //opencv-affine-transform
            t4cl::confirmAffine(gray, affineTransformed, scaleX, scaleY, transX,
transY, deg);
        }
            std::cout << "\r" << "complete affineTransform" << std::endl;
        if (num != 0) break;
    case 4:
        std::cout << "run templateMatching";
        {
            cv::Mat1b result, resizedGray;
            cv::resize(gray, resizedGray, 0.5, 0.5);
            cv::Mat1b temp(resizedGray, cv::Rect(75, 75, 15, 15));
            cv::Rect matchingArea;
            uchar color(0);
            // src template dstRect
            t4cl::templateMatching(resizedGray, temp, matchingArea);
            t4cl::drawRect(gray, result, matchingArea, color);
            cv::imshow("template", temp);
            cv::moveWindow("template", 100 + result.cols, 100);
            t3cl::imshow("templateMatching", result);
        }
            std::cout << "\r" << "complete templateMatching" << std::endl;
        if (num != 0) break;
    case 5:
        std::cout << "run bilateral";
        {
            cv::Mat1b result, area(gray);
            // src dst kernelsize standardDeviation1 standardDeviation2
            t4cl::bilateral(gray, result, 5, 30, 30);
            t3cl::imshow("bilateral", result);
        }
            std::cout << "\r" << "complete bilateral" << std::endl;
        if (num != 0) break;
    case 6:
        std::cout << "run thinning";
        {
            cv::Mat1b thinned;
            t4cl::thinning(hirakawa, thinned);
            t3cl::imshow("thinned", thinned);
        }
            std::cout << "\r" << "complete thinning" << std::endl;
        if (num != 0) break;
    }
}
```