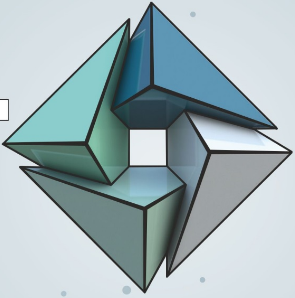
ZT  RW

Digital image processing #3

Geometric transformations

Kamil Szeląg

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

December 13, 2017

Geometric transformations

Warp affine

- Task #27
  - Load color image
  - Create result image
  - Create 2x3 floating point matrix representing identity transform
  - Apply warp affine - what are differences in opencv implementation
- Task #28
  - Create translation matrix and apply translation on loaded image. What does image size parameter mean in warp affine matrix.
- Task #29
  - Create rotation matrix and apply rotation on loaded image. What does image size parameter mean in warp affine matrix.

Geometric transformations

Warp affine

```
//Task #27
        cv::Mat img = cv::imread("lena.bmp");
        cv::Mat result = cv::Mat::zeros(img.size(), img.type());

        cv::Mat transformation = (cv::Mat_<double>(2, 3) << 1, 0, 0, 0, 1, 0);
        cv::warpAffine(img, result, transformation, result.size());

//Task #28
        cv::Mat img = cv::imread("lena.bmp");
        cv::Mat result = cv::Mat::zeros(img.size(), img.type());

        cv::Mat transformation = (cv::Mat_<double>(2, 3) << 1, 0, 100, 0, 1, 100);
        cv::warpAffine(img, result, transformation, result.size());
        cv::warpAffine(img, result, transformation, result.size()*2);

//Task #29
        cv::Mat img = cv::imread("lena.bmp");
        cv::Mat result = cv::Mat::zeros(img.size(), img.type());

        cv::Mat transformation = cv::getRotationMatrix2D(cv::Point2f(img.cols / 2,
                                                          img.rows / 2), 30,1);
        cv::warpAffine(img, result, transformation, result.size());
```

Complex transformations

- Task #30
  - Load image
  - Create translation matrix with arbitrary translation.
  - Create rotation matrix with arbitrary rotation.
  - Perform two transformations. Is it possible to compose matrices into single one?

Geometric transformations

Complex transformations

```
//Task #30
        cv::Mat img = cv::imread("E:\\Workspace\\KS\\TestData\\sam_cam\\1.jpg");
        cv::Mat result, result2;
        cv::Point2f center =cv::Point2f( result.cols / 2, result.rows / 2 );
        cv::Mat translation = (cv::Mat_<double>(2, 3) << 1, 0, 100, 0, 1, 100);
        cv::Mat rotation = cv::getRotationMatrix2D(center,30,1);
        cv::warpAffine(img, result, rotation, result.size());
        cv::warpAffine(result, result, translation, result.size());

        cv::Mat row = (cv::Mat_<double>(1, 3) << 0, 0, 1);
        cv::Mat translation_ex = translation.clone();
        translation_ex.push_back(row);

        cv::Mat rotation_ex = rotation.clone();
        rotation_ex.push_back(row);

        cv::Mat transformation = translation_ex*rotation_ex;
        cv::warpAffine(img, result2, transformation(cv::Rect(0,0,3,2)), result.size());
```

Geometric transformations

How to invert transformation?

- Task #31
  - Load color image
  - Define arbitrary affine matrix and perform image transformation
  - Calculate and perform reverse image transformation.

Geometric transformations

How to invert transformation?

```
//Task #31
    cv::Mat img = cv::imread("E:\\Workspace\\KS\\TestData\\sam_cam\\1.jpg");
    cv::Mat result, result2;
    cv::Point2f center =cv::Point2f( result.cols / 2, result.rows / 2 );
    cv::Mat translation = (cv::Mat_<double>(2, 3) << 1, 0, 100, 0, 1, 100);
    cv::Mat row = (cv::Mat_<double>(1, 3) << 0, 0, 1);
    cv::Mat translation_ex = translation.clone();
    translation_ex.push_back(row);

    cv::Mat inverse;
    inverse = translation.inv();
    cv::warpAffine(img, result2, inverse(cv::Rect(0, 0, 3, 2)), result.size());
```

Geometric transformations

Get affine transformation

- Task #32
  - Create image with rectangle
  - Pick three points on that rectangle
  - Pick three destination points of rectangle
  - Get affine transformation between these two views in two different ways (one with solving set of equations)

Geometric transformations

Get affine transformation

```
//Task #32
    cv::Mat img = cv::Mat::zeros(500, 500, CV_8U);
    cv::rectangle(img, cv::Point(50, 50), cv::Point(250, 250), cv::Scalar::all(255), 3);

    std::vector<cv::Point2f>src_points{ cv::Point2f(50,50),cv::Point2f(250,50) ,
 cv::Point2f(250,250) };
    std::vector<cv::Point2f>dst_points{ cv::Point2f(50,50),cv::Point2f(250,125) ,
 cv::Point2f(125,250) };

    //opencv way
    cv::Mat T = cv::getAffineTransform(src_points, dst_points);

    cv::Mat src_mat = cv::Mat(src_points.size(), 2, CV_32F, src_points.data());
    src_mat = src_mat.t();
    cv::Mat dst_mat = cv::Mat(dst_points.size(), 2, CV_32F, dst_points.data());
    dst_mat = dst_mat.t();


    cv::Mat row = (cv::Mat_<float>(1, 3) << 1, 1, 1);

    src_mat.push_back(row);
    dst_mat.push_back(row);

    // dst = T * src
    // T  = dst* src_inv
    cv::Mat T_eq = dst_mat*src_mat.inv();
```

Geometric transformations

Get perspective transformation

- Task #33
  - Create image with rectangle
  - Pick four points on that rectangle
  - Pick four destination points of rectangle
  - Get perspective transformation between these two views.

Geometric transformations

Get affine transformation

```
//Task #32
        cv::Mat img = cv::Mat::zeros(500, 500, CV_8U);
        cv::rectangle(img, cv::Point(50, 50), cv::Point(250, 250), cv::Scalar::all(255), 3);

        std::vector<cv::Point2f>src_points{ cv::Point2f(50,50),
    cv::Point2f(250,50) ,cv::Point2f(250,250),
    cv::Point2f(50,250) };
        std::vector<cv::Point2f>dst_points{ cv::Point2f(50,50),
    cv::Point2f(250,125) ,cv::Point2f(125,250),
    cv::Point2f(125,50) };

        cv::Mat T = cv::getPerspectiveTransform(src_points, dst_points);
```

The end

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