unesco
University Network
Member

Ministry of
Education

2D Projective Transformation

Sung Soo Hwang

Perspective Transformation

- getPerspectiveTransform() function

◆ getPerspectiveTransform() [1/2]

```
Mat cv::getPerspectiveTransform ( const Point2f  src[],
                                  const Point2f  dst[]
                                )
```

  - returns 3x3 perspective transformation for the corresponding 4 point pairs.

3x3 perspective matrix
(8 DOF)

$$\begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

dst　　src

Perspective Transformation

- wrapPerspective() function

◆ warpPerspective()

```
void cv::warpPerspective ( InputArray    src,
                           OutputArray   dst,
                           InputArray    M,
                           Size          dsize,
                           int           flags = INTER_LINEAR ,
                           int           borderMode = BORDER_CONSTANT ,
                           const Scalar & borderValue = Scalar()
                         )
```

  - src: input image.
  - dst: output image that has the size dsize and the same type as src.
  - M: 3 X 3 transformation matrix.
  - dsize: size of the output image.

Perspective Transformation

- wrapPerspective() function

◆ warpPerspective()

```
void cv::warpPerspective ( InputArray    src,
                           OutputArray   dst,
                           InputArray    M,
                           Size          dsize,
                           int           flags = INTER_LINEAR ,
                           int           borderMode = BORDER_CONSTANT ,
                           const Scalar & borderValue = Scalar()
                         )
```

  - Applies a perspective transformation to an image.
  - The function warpPerspective transforms the source image using the specified matrix:

$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right)$$

Perspective Transformation

- wrapPerspective() function
  - flag: combination of interpolation methods(INTER_LINEAR or INTER_NEAREST) and the optional flag WRAP_INVERSE_MAP, that sets M as the inverse transformation(dst → src).
  - borderMode: pixel extrapolation method (BORDER_CONSTANT or BORDER_REPLICATE).
  - borderValue: value used in case of a constant border; by default, it equals 0.

Perspective Transformation

- Example code

```
struct MouseParams
{
   Mat img;
   vector<Point2f> in, out;
};


static void onMouse(int event, int x, int y, int, void* param)
{
   MouseParams* mp = (MouseParams*)param;
   Mat img = mp->img;
   if (event == EVENT_LBUTTONDOWN) // left button
   {
      Mat result;

      //Insert position from LT. Direction is clock-wise
      mp->in.push_back(Point2f(x, y));


      if (mp->in.size() == 4)
      {
         // Calculate perspective transform matrix(=homo_mat) from 4 matching pairs of points
         Mat homo_mat = getPerspectiveTransform(mp->in, mp->out);
```

Perspective Transformation

- Example code

```
// apply perspective transformation to img using homo_mat
        // result will have the same size of Size(300, 300) and the same type of img
        warpPerspective(img, result, homo_mat, Size(300, 300));
        imshow("output", result);
    }
    else
    {
        result = img.clone();
        for (size_t i = 0; i < mp->in.size(); i++)
        {
            circle(result, mp->in[i], 3, Scalar(0, 0, 255), 5);
        }
        imshow("input", result);
    }
}

//Reset positions
if (event == EVENT_RBUTTONDOWN)
{
    mp->in.clear();
    imshow("input", img);
}
}
```

Perspective Transformation

- Example code

```
int main()
{
    Mat imput = imread("book.jpg");
    imshow("input", imput);

    MouseParams mp;
    mp.out.push_back(Point2f(0, 0));
    mp.out.push_back(Point2f(300, 0));
    mp.out.push_back(Point2f(300, 300));
    mp.out.push_back(Point2f(0, 300));
    mp.img = imput;

    setMouseCallback("input", onMouse, (void*)&mp);
    waitKey();
    return 0;
}
```

Perspective Transformation

- Result