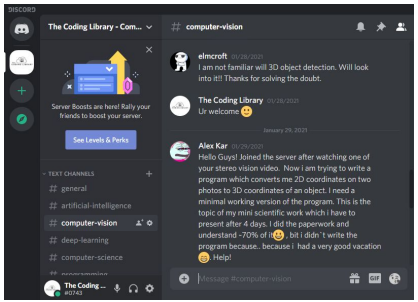
THE CODING LIBRARY

Computer Vision

Geometric Transformations

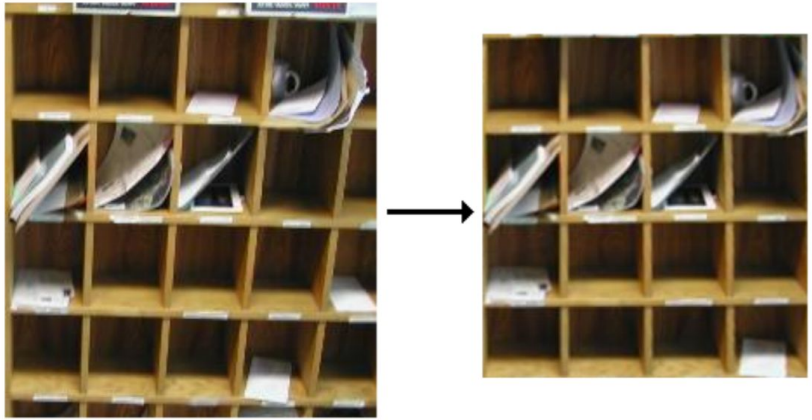


Discord Link in Description

Geometric Transformations

THE CODING LIBRARY

- Types of Geometric Transformations:
  - Geometric
  - Pixel coordinate
  - Brightness interpolation
- Applications:
  - Computer graphics
  - Distortion - Introduce / Eliminate
  - Image processing / Preprocessing
  - Text recognition
  - Recognition of signs, numbers, etc.

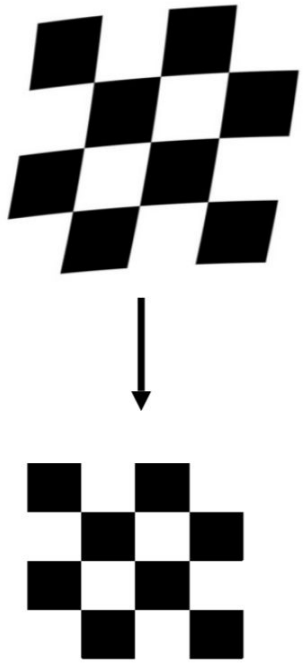


Formulation of the problem

- Distorted image $f(i,j)$
- Corrected image $f'(i',j')$
- Mapping between the images

$$i = T_i(i',j') \qquad j = T_j(i',j')$$

- Define the Transformation
  - Known in advance / determine through correspondences
  - Image to know
  - Image to image
- Apply the defined Transformation
  - For every point in the output image
  - Determine where it came from using T from mapping
  - Interpolate a value for the output point

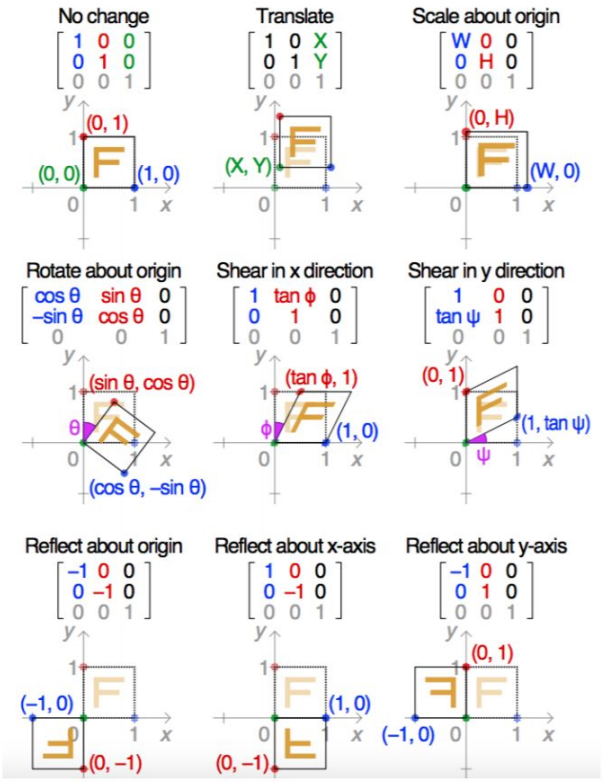


THE CODING LIBRARY

Transformations

- Linear Transformations in 3D
  - Affine Transformation
    - Euclidean (6 DoF)

- Higher Order Transformations
  - Parameterized
    - B-splines
  - Freeform
    - Warp field



Affine Transformations in 2D

Affine Transformations

THE CODING LIBRARY

- Known Transformations
  - E.g. Translation, Rotation, etc
- Unknown Transformations
  - Would require at least 3 observations
  - Could be points in an image
    - E.g. Cornes of objects, etc.

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

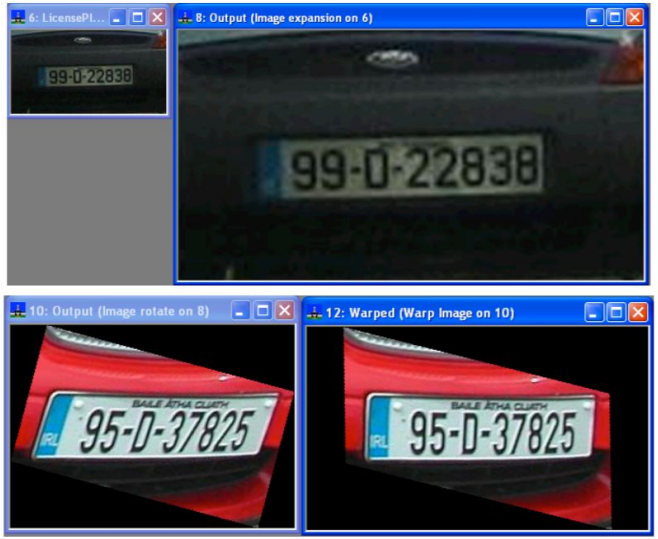
$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 1 & 0 & m \\ 0 & 1 & n \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

```
Mat affine_matrix( 2, 3, CV_32FC1 );
…
warpAffine( image, result, affine_matrix, image.size() );
```

Example of a Simple Affine Transformation

THE CODING LIBRARY

1. Rotating the image
2. Scaling the image
3. Skewing or shearing the image



Unknown Affine Transformation

THE CODING LIBRARY

- More observations - We need at least 3
  - Better estimate of the coefficients
- Uses pseudo inverse
  - For unknown transformations

```
Point2f source [3], destination [3];
…
affine_matrix = getAffineTransform(source,destination);
```

Perspective Transformations

THE CODING LIBRARY

- Perspective projection
- Planar surface
- Not parallel to the image plane
- Can’t be corrected with the affine trans
- Therefore we need a perspective trans

$$\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} i.w \\ j.w \\ w \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & 1 \end{bmatrix} \begin{bmatrix} i' \\ j' \\ 1 \end{bmatrix}$$

Perspective Transformations in OpenCV

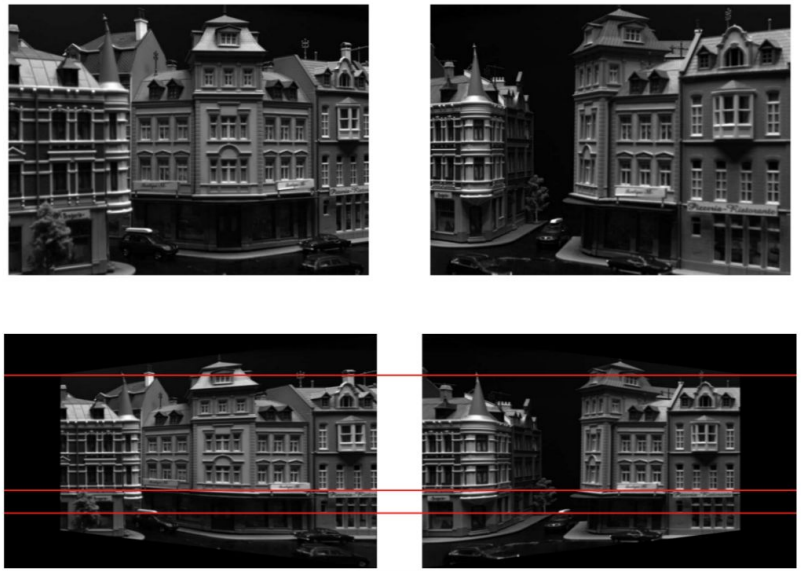
THE CODING LIBRARY

```
Point2f source [4], destination [4];
// Assign values to source and destination points.
perspective_matrix = getPerspectiveTransform( source, destination );
warpPerspective( image, result, perspective_matrix,
result.size() );
```

Rectifying Homographies

THE CODING LIBRARY

- Image transformation can be computed such that scanlines can be directly matched on images

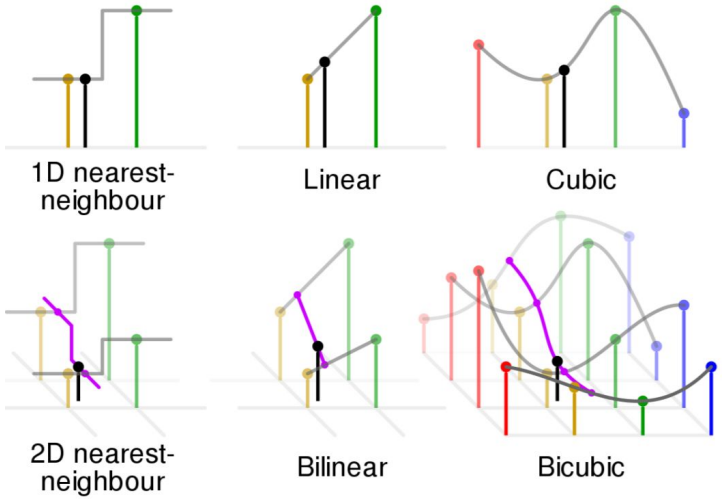


Brightness Interpolation

THE CODING LIBRARY

- Locations are not integer coordinates
- Interpolate output pixel value from the nearby pixels in the original image

- Interpolation methods
  - Nearest neighbour
  - Bilinear interpolation
  - Bicubic interpolation



Brightness Interpolation in OpenCV

THE CODING LIBRARY

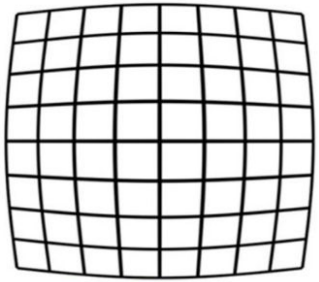
```
  int interpolation_scheme = INTER_LINEAR;
  // Nearest neighbour interpolation: INTER_NEAREST
  // Bilinear interpolation: INTER_LINEAR
  // Bicubic interpolation: INTER_CUBIC
  warpAffine( image, result, affine_matrix, result_size,

interpolation_scheme );
  warpPerspective( image, result, perspective_matrix,
                                          result_size,
interpolation_scheme );
```

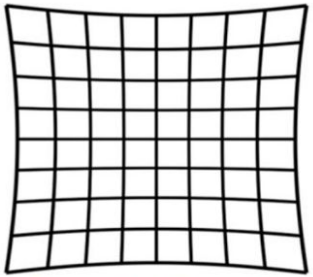
Distortion - Camera Models

THE CODING LIBRARY

- Radial Distortion
  - Radially symmetric from the optical axis
  - Change in magnification
  - Barrel distortion
  - Pincushion distortion
  - Caused by the lenses

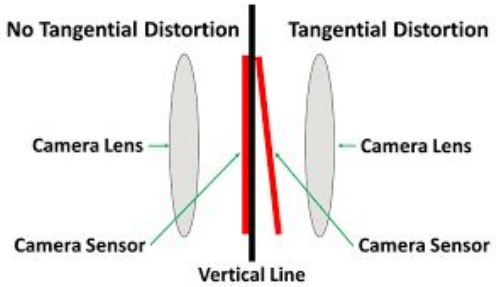


Barrel Distortion



Pincushion Distortion

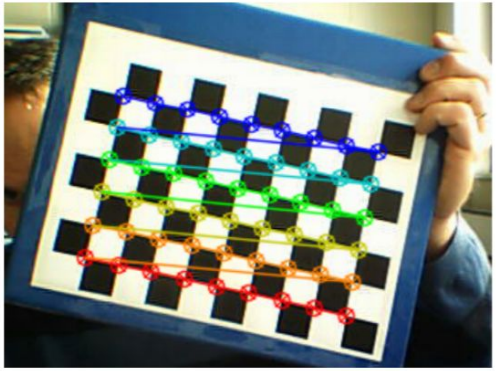
- Tangential Distortion
  - Uneven magnification from one side to the other
  - Lenses not parallel to the image plane



Removing Distortion - Camera Models

THE CODING LIBRARY

- Calibrate using multiple images
- Calibrate with known objects
- Change positions
- Compute the camera matrix and distortion parameters
- Remove distortion using the distortion parameters



```
calibrateCamera( object_points, image_points,
      image_size, camera_matrix, distortion_coefficients,
      rotation_vectors, translation_vectors );
…
 undistort( camera_image, corrected_image,
              camera_matrix, distortion_coefficients );
```