CV Practice Class
Lecture 3

2017-07-05

FIRA 인공지능 에이전트 과정
SNUVL Lab

Contents

Update Practice Git Repo.

- Git pull
  - At the `fira-2017-cv` root folder, get the last commits via `git pull`

```
$ git pull
```

- In case you get error such as

```
error: cannot pull with rebase: You have unstaged changes
error: please commit or stash them.
```

  - First, you should enter your name and email address(for the first time),

```
$ git config --global user.name "YOUR NAME"
$ git config --global user.email "YOUR@EMAIL.com"
```

  - Then you should `stash` the changes, `pull` last commits, and `stash pop`.

```
$ git stash
$ git pull
$ git stash pop
```

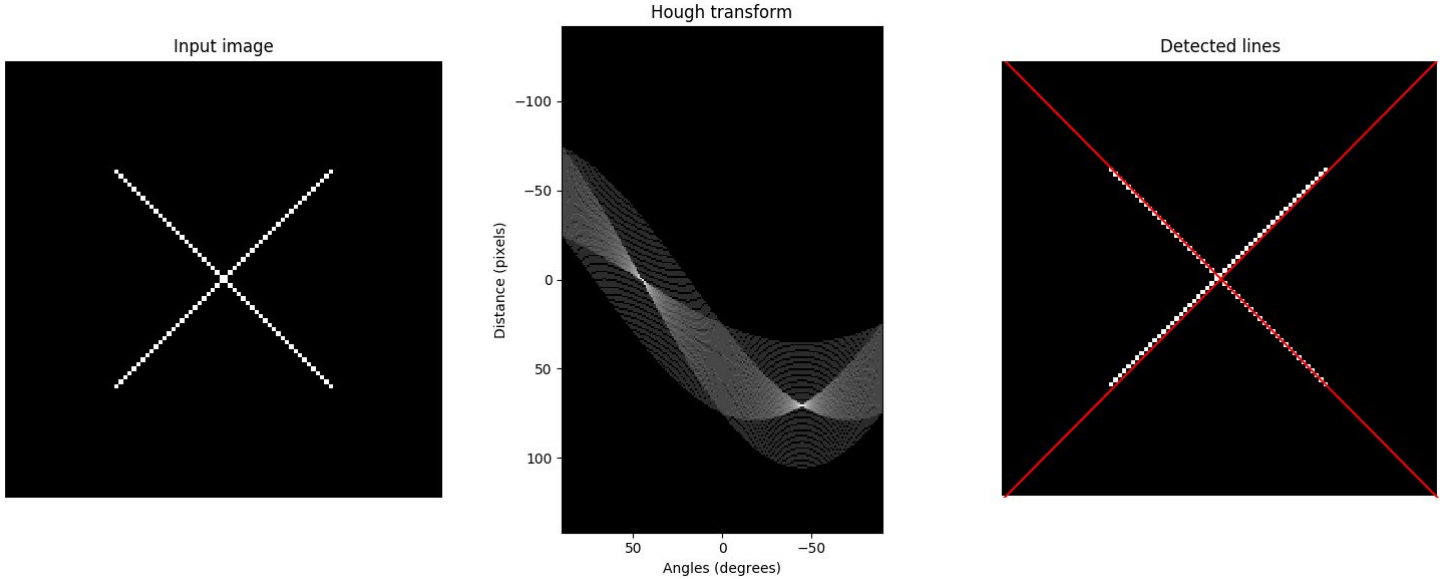
Hough Transform

Introduction
Hough Line Transform
Hough Circle Transform

Content credit: Wikipedia, OpenCV tutorials

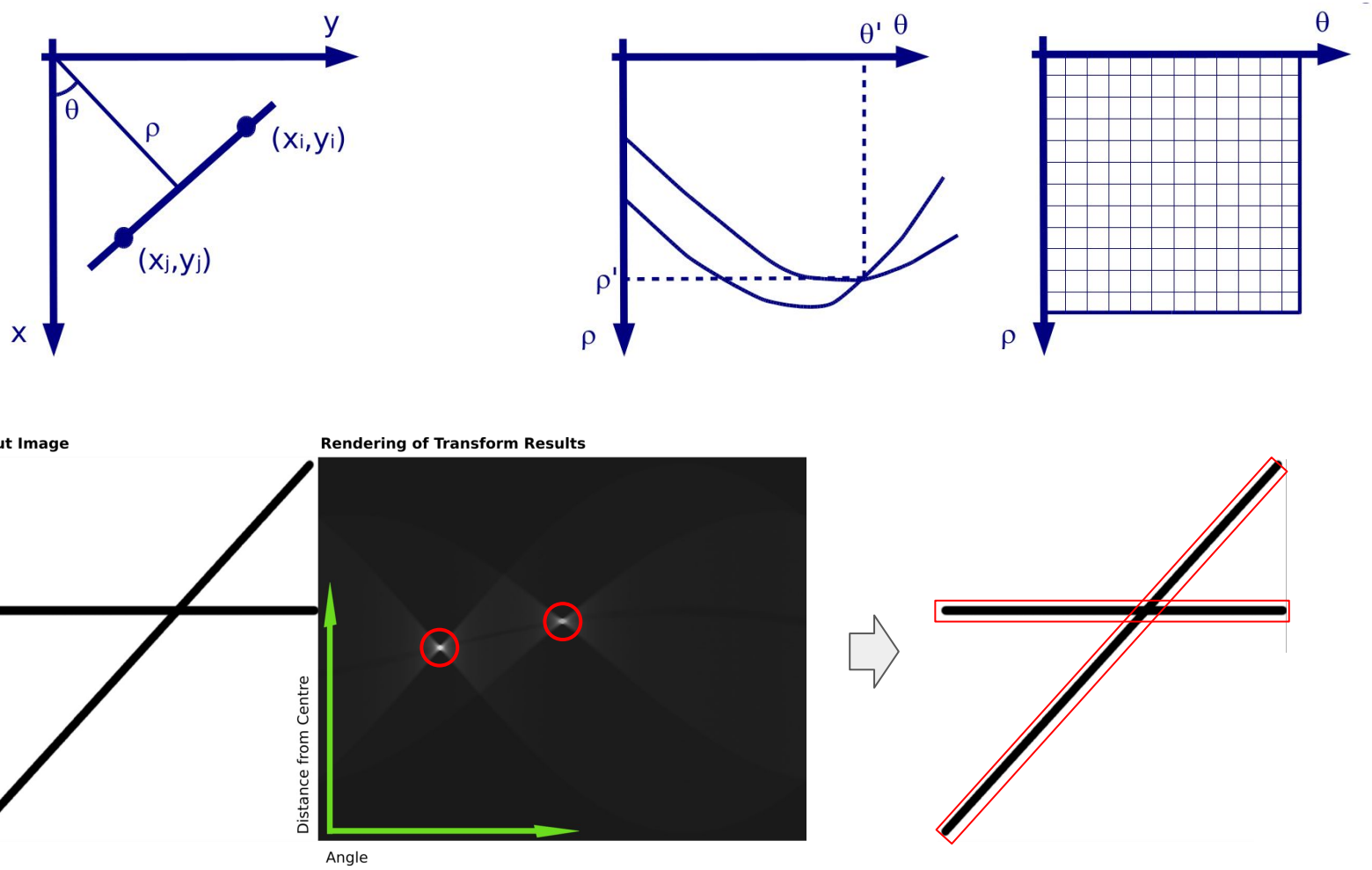
Hough Transform

- Hough Transform
  - A feature extraction technique used to **find imperfect instances of objects**.
  - Transformation? - It convert an image to some **parameter space** by a voting process, and find objects by **finding maxima** in that sapce.
  - What parameters? - Any parameters that can represent a **certain class of shapes**(of interest; lines, circles, ...)
  - How? - For each pixels, **vote for the parameters** of which the pixel can be part of the shapes.



Hough Transform

- Hough Line Transform
  - Voting process for Hough line transform(ϱ(rho) and θ(theta))



Hough Transform

- Hough line transform

```
lines = cv2.HoughLines(image, rho, theta, threshold, lines=None,
                       srn=0, stn=0, min_theta=0, max_theta=np.pi)
```

- Inputs
  - `image`: A 8-bit, single-channel input image. Usually the output of edge detection to find lines as edges
  - `rho`: The histogram bin size in rho(ρ) dimension
  - `theta`: The histogram bin size in theta(θ) dimension
  - `threshold`: The threshold of accumulator for detecting a line
  - `lines`: The output lines, if you have allocated beforehand.
  - `srn`: A divisor for the rho, for the multi-scale Hough transform
  - `stn`: A divisor for the theta, for the multi-scale Hough transform
  - `min_theta`: Minimum angle to check for lines
  - `max_theta`: Maximum angle to check for lines
- Output
  - `lines`: Array of line parameters(rho, theta)

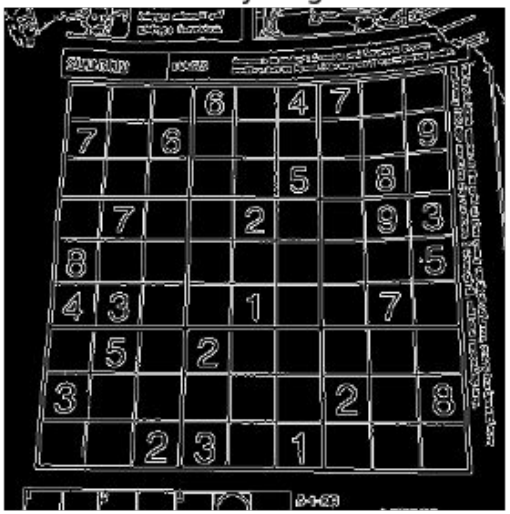
Hough Transform

- Hough line transform

```
img = cv2.imread('sudoku.jpg', cv2.IMREAD_COLOR)
canny_edges = cv2.Canny(img, 50, 100)
lines = cv2.HoughLines(canny_edges, 1, np.pi/180, 200)
```

Input image

Canny edge



Hough transform output

- Note that the lines are drawn via `cv2.line()` function given the parameters of lines(rho, theta).

Hough Transform

- Hough circle transform

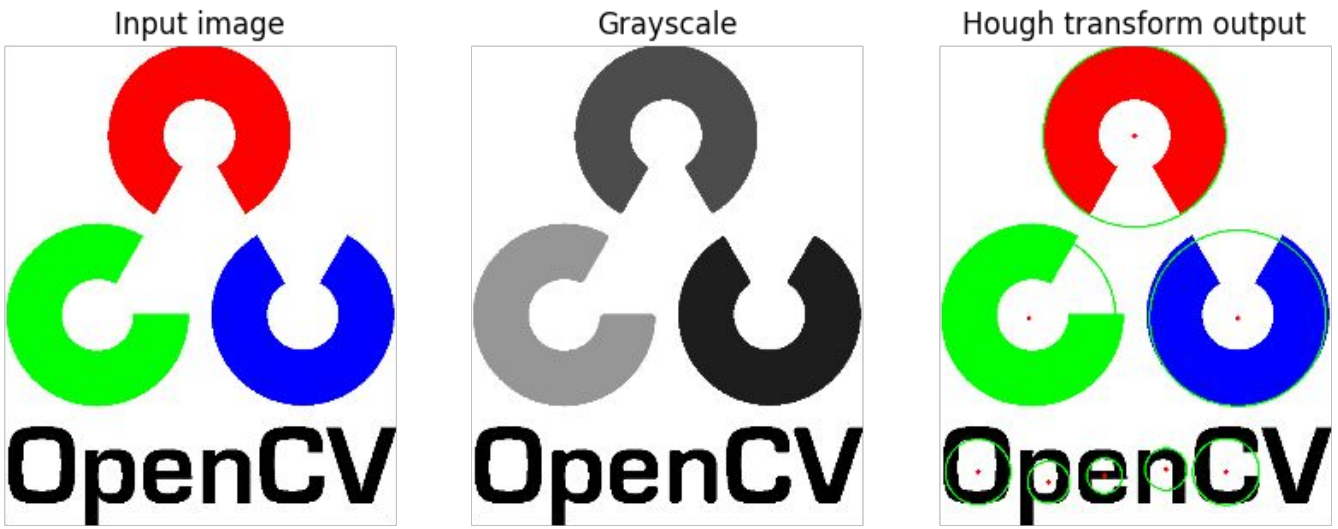
```
circles = cv2.HoughCircles(image, method, dp, minDist, circles=None,
                           param1=, param2=, minRadius=, maxRadius=)
```

- Input
  - `image`: A 8-bit, single-channel input image
  - `method`: The method to perform Hough circle transform, Currently, only `cv2.HOUGH_GRADIENT` is available.
  - `dp`: Inverse ratio of the accumulator resolution to the image resolution.
  - `minDist`: Minimum distance between the centers of circles.
  - `circles`: The output circle, if you have allocated beforehand.
  - `param1`: First parameter. If method is cv2.HOUGH_GRADIENT, it is the higher threshold for canny edge detector(the lower one is twice smaller).
  - `param2`: Second parameter. If method is cv2.HOUGH_GRADIENT, it is the accumulator threshold for the circle centers at the detection stage.
  - `minRadius`: Minimum circle radius / `maxRadius`: Maximum circle radius
- Output
  - `circles`: Array of circle parameters(center_x, center_y, radius)

Hough Transform

- Hough circle transform

```
img = cv2.imread('opencv_logo2.png', cv2.IMREAD_COLOR)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGRA2GRAY)
circles = cv2.HoughCircles(img_gray, cv2.HOUGH_GRADIENT, 1, 20,
                    param1=120, param2=35, minRadius=0, maxRadius=0)
```



- Note that since the only method used is **Hough Gradient Method**, so the canny edge detection is performed inside `cv2.HoughCircles()` function.
- It is preferred to blur the input image before to reduce the noise.

Hough Transform

- Practice: Coin Removal
  - Given image of coins on a table, detect the coins and remove them
  - cv2.HoughCircles() for detecting circles
  - cv2.circle() for filling the circle with the background color

Let’s Check the Code

`1_hough_transform.ipynb`

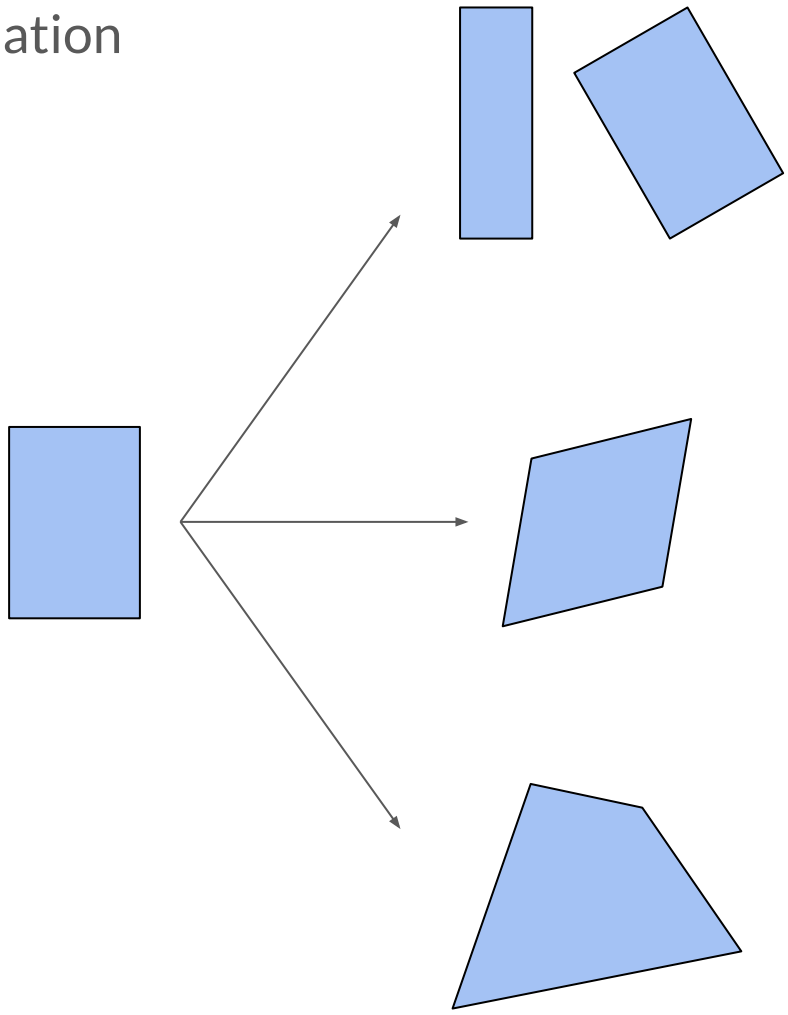
Geometric Image Transformation

Introduction
Applying Transformation
Getting Transformation Matrix

Content credit: Alyosha Efros, OpenCV tutorials

Geometric Transformation

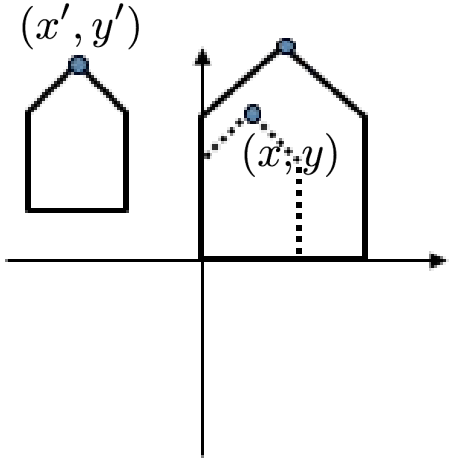
- Similarity Perspective Transformation
  - Scale / Translation / Rotation
- Affine Transformation
- Perspective Transformation
  - Homograpy



Geometric Transformations

- Scale

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

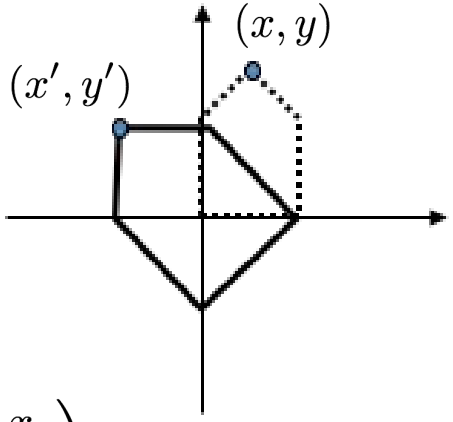


- Translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Rotation
  - With no scale, translation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



  - With scale, translation(rotation center on (Cx, Cy))

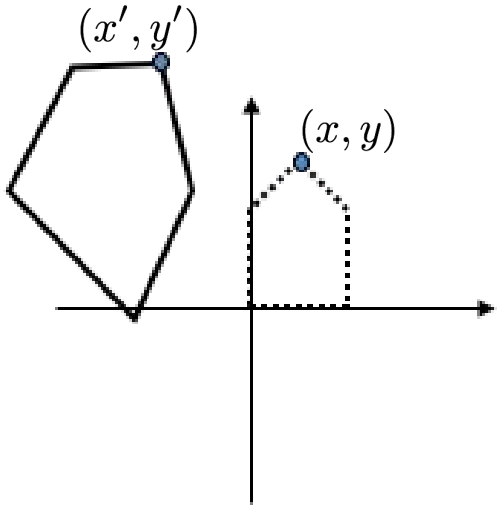
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S\cos\theta & -S\sin\theta & C_x(1-\cos\theta) + C_y\sin\theta \\ S\sin\theta & S\cos\theta & C_x\sin\theta + C_y(1-\cos\theta) \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Geometric Transformations

- Affine Transformation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b & T_x \\ c & d & T_y \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

  - 6 free parameters



- Perspective Transformation

$$\begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \begin{aligned} x' &= u'/w' \\ y' &= v'/w' \end{aligned}$$

  - 8 free parameters

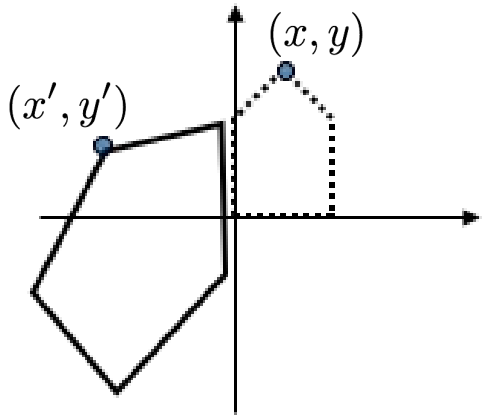
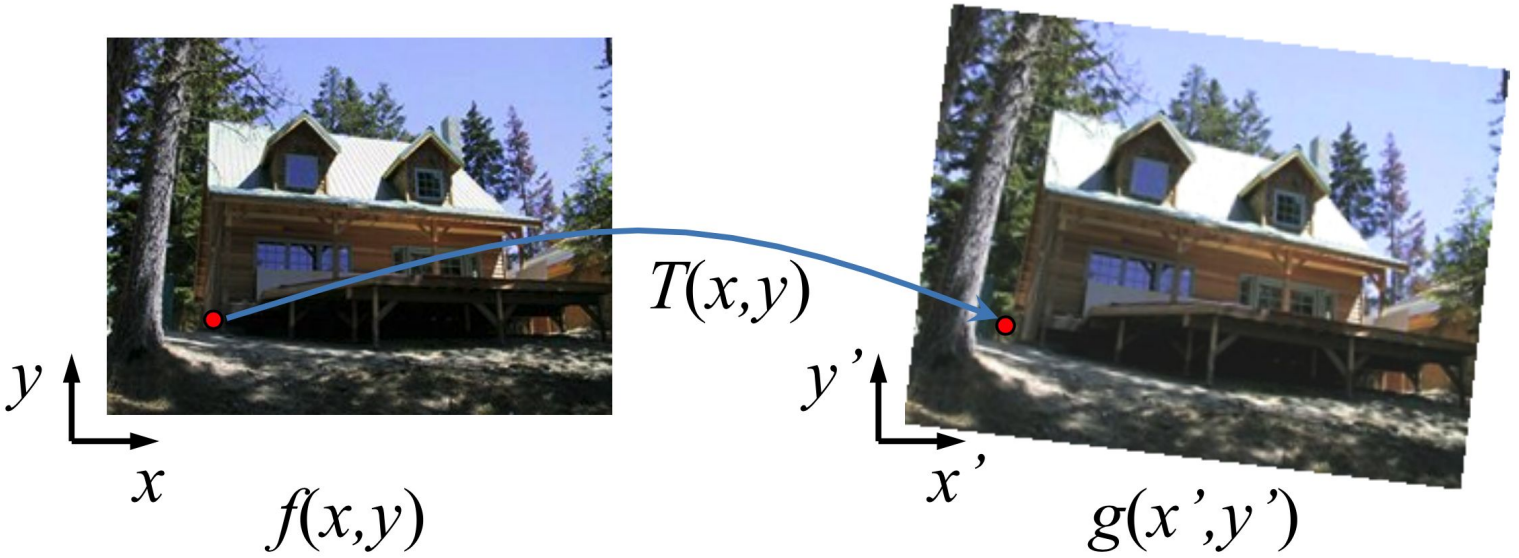


Image Warping

- Image Warping
  - Given a coordinate transform T and a source image $f(x,y)$, finding a transformed image which satisfies $g(x',y') = f(T(x,y))$.

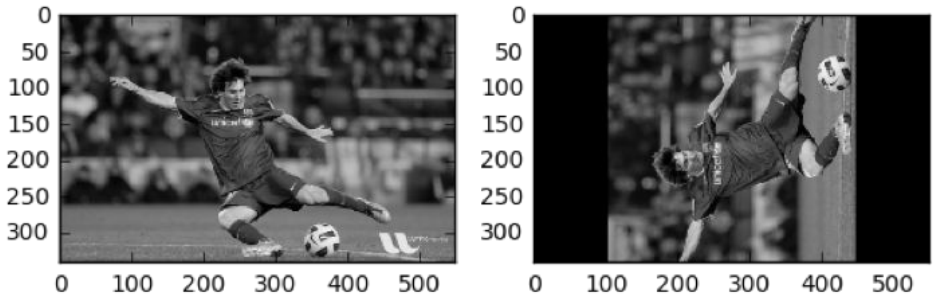


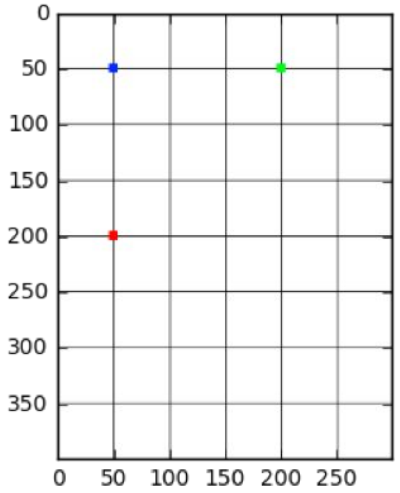
Geometric Transformation

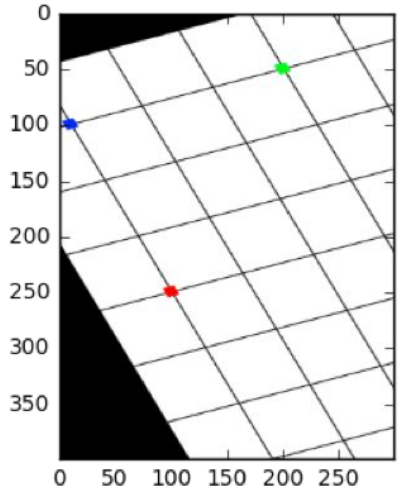
- Apply Affine Transformation

```
pts2 = cv2.transform(pts1, M, ...) # points
dst = cv2.warpAffine(src, M, dsize, dst, flags=cv2.CV_INTER_LINEAR+cv2
.CV_WARP_FILL_OUTLIERS, borderMode=cv2.BORDER_CONSTANT, borderValue=0)
# image
```

  - `M`: Affine matrix(2×3)
  - `flags`: Interpolation methods and optional flag `cv2.WARP_INVERSE_MAP`
  - `borderMode`: Pixel extrapolation method





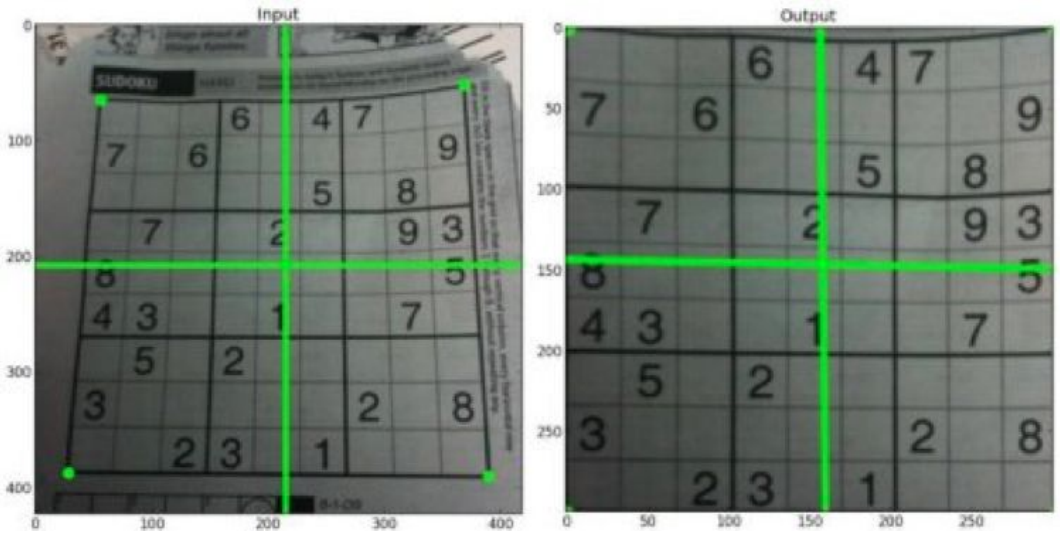


Geometric Transformation

- Apply Perspective Transformation

```
pts2 = cv2.perspectiveTransform(pts1, M) # points
dst = cv2.warpPerspective(src, M, dsize, dst, flags=cv2.CV_INTER_LINEAR+cv2.CV_WARP_FILL_OUTLIERS, borderMode=cv2.BORDER_CONSTANT, borderValue=0) # image
```

  - `M`: Homography matrix(3×3 with `M[3,3]==1`)
  - `flags`: Interpolation methods and optional flag `cv2.WARP_INVERSE_MAP`
  - `borderMode`: Pixel extrapolation method



Geometric Transformation

- Apply Perspective Transformation
  - `borderMode`: Pixel extrapolation method

Original Image

borderMode=
cv2.BORDER_CONSTANT

borderMode=
cv2.BORDER_WRAP

borderMode=
cv2.BORDER_REFLECT

Geometric Transformation

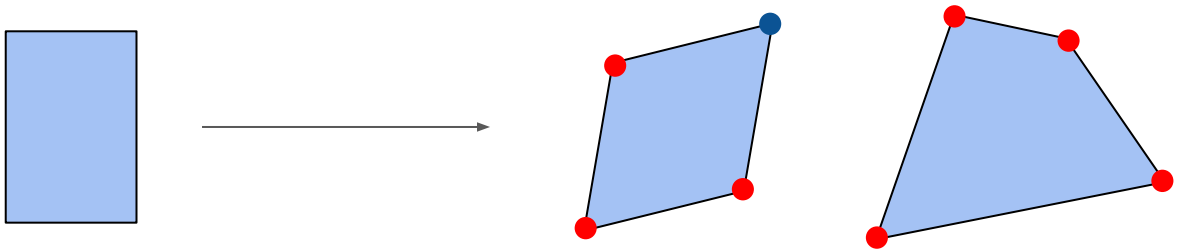
- Get Transformation Matrix

```
M = cv2.getRotationMatrix2D(center, angle, scale)
```

  - `center`: Center of the rotation in the source image.
  - `angle`: Rotation angle in degrees.
  - `scale`: Isotropic scale factor.

```
M = cv2.getAffineTransform(src, dst)
M = cv2.getPerspectiveTransform(src, dst)
```

  - `src`: Coordinates of triangle/quadrangle vertices in the source image.
  - `dst`: Coordinates of the corresponding triangle/quadrangle vertices in the destination image.
    (`src`, `dst` are N×2(N: 3 for affine or 4 for perspectice) NumPy matrices)



Geometric Transformation

- Practice #1: Track Advertisement

Logo

Track

Overlayed logo

- Practice #2: Cam Scanner

Input image

Sides

Corners

Output

Let’s Check the Code

`2_geometric_transform.ipynb`