

# Transformaciones

Cátedra Visión por Computadoras

13 de abril de 2022

- Usamos la función  
`dst = cv2.warpAffine(src, M, dsize[, dst])`
- Es una función genérica que nos permite aplicar una matriz de transformación **M** de tamaño  $2 \times 3$ 
  - ▶ **src** es la imagen a transformar
  - ▶ **M** es la matriz de transformación, ver más abajo
  - ▶ **dsize** es el tamaño de la imagen de salida
  - ▶ **dst** imagen de salida que le podemos pasar en forma opcional, caso contrario es el valor de retorno de la función

En el caso de una **traslación** la matriz **M** nos queda:

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \end{bmatrix}$$

## Código para realizar una traslación

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2

def translate(image, x, y):
    (h, w) = (image.shape[0], image.shape[1])

    M = np.float32([[1, 0, x],
                    [0, 1, y]])

    shifted = cv2.warpAffine(image, M, (w, h))

    return shifted
```

## Rotación

- Primero usamos el método **M** = `cv2.getRotationMatrix2D(center, angle, scale)` para calcular la matriz de rotación
  - ▶ **center** es el centro de rotación en la imagen de entrada, (**x**, **y**)
  - ▶ **angle** es el ángulo de rotación, entendido en sentido antihorario y suponiendo como origen la coordenada superior izquierda
  - ▶ **scale** factor de escala isotrópico
  - ▶ **M** es la matriz de rotación que nos devuelve el método
- Y luego usamos esa matriz obtenida con el método `cv2.warpAffine`

En el caso de la rotación la matriz **M** nos queda:

$$\begin{bmatrix} s \cdot \cos(\text{angle}) & s \cdot \sin(\text{angle}) & [1-s \cdot \cos(\text{angle})]x - s \cdot \sin(\text{angle}) \cdot y \\ -s \cdot \sin(\text{angle}) & s \cdot \cos(\text{angle}) & s \cdot \sin(\text{angle}) \cdot x - (1-s \cdot \cos(\text{angle})) \cdot y \end{bmatrix}$$

donde *s* es el factor de escala y (*x*, *y*) son las coordenadas del centro de rotación.

## Código para realizar una rotación

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2

def rotate(image, angle, center=None, scale=1.0):
    (h, w) = image.shape[:2]

    if center is None:
        center = (w/2, h/2)

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

    rotated = cv2.warpAffine(image, M, (w, h))

    return rotated
```

# Espejado

- Usamos la función  
`dst = cv2.flip(src, flipCode[, dst])`
- **src** es la imagen a transformar
- **flipCode** es un entero que indica la forma del espejado:
  - ▶ 0 indica espejado sobre el eje **x**
  - ▶ 1 indica espejado sobre el eje **y**
  - ▶ -1 indica espejado sobre ambos ejes
- **dst** imagen de salida que le podemos pasar en forma opcional, caso contrario es el valor de retorno de la función

# Código para realizar un espejado

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import cv2

modes = {'x': 0, 'y': 1, 'b': -1}

def flip(img, mode):
    if mode not in modes.keys():
        return img

    flipped = cv2.flip(img, modes[mode])

    return flipped
```

## Prácticos

Esta unidad tiene asociado los prácticos 5, 6, 7 y 8.