ЗАПИСКИ ПРЕПОДАВАТЕЛЯ

Ich sage euch: man muß noch Chaos in sich haben, um einen tanzenden Stern gebären zu können (“Also sprach Zarathustra”, Friedrich Nietzsche)

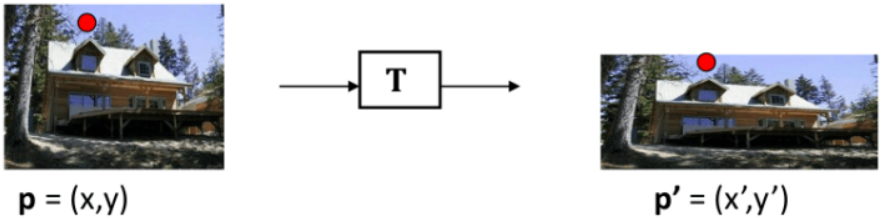
ОБРАБОТКА ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ OPENCV В PYTHON

Познайте методы предварительного преобразований плоского изображения, такие как перемещение

изображения, отражение, поворот, масштабирование, обрезка и нарезка с помощью библиотеки OpenCV в Python.

ВВЕДЕНИЕ

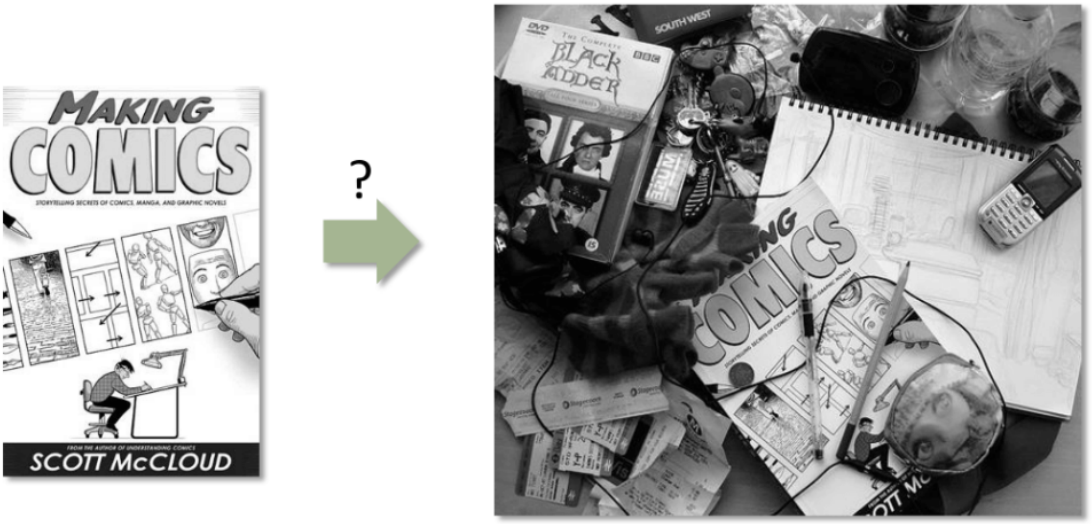
По существу, трансформация изображения — это его отображение из одной системы координат в другою, она сопоставляет некоторые координаты точки (x, y) в одной системе с точкой координатами (x', y') в другой системе координат.



Например, если у нас есть точка с координатами $(2, 3)$ в системе координате $x \times y$ и мы строим ту же точку в системе координате $u \times v$, естественно, она будет выглядеть по-разному, как и показано на рисунке ниже:

ДЛЯ ЧЕГО НЕОХОДИМО ПРЕОБРАЗОВЫВАТЬ ИЗОБРАЖЕНИЯ?

На изображении ниже геометрическое соотношение между комиксом и изображением справа основано на преобразовании подобия (поворот, перемещение и масштабирование). Если нам нужно обучить модель машинного обучения, которая находит этот комикс, то нам нужно ввести изображение в другой форме и под другим углом.



Методы преобразования изображений очень нам помогут на этапе предварительной обработки изображений в машинном обучении.

Изображение может быть представлено как матрица. Каждое значение в матрице — это цвет пикселя в определенной координате. Преобразование изображения может быть выполнено с использованием матричного умножения. В математике есть несколько матричных уравнений, которые можно использовать для выполнения определенных операций преобразования.

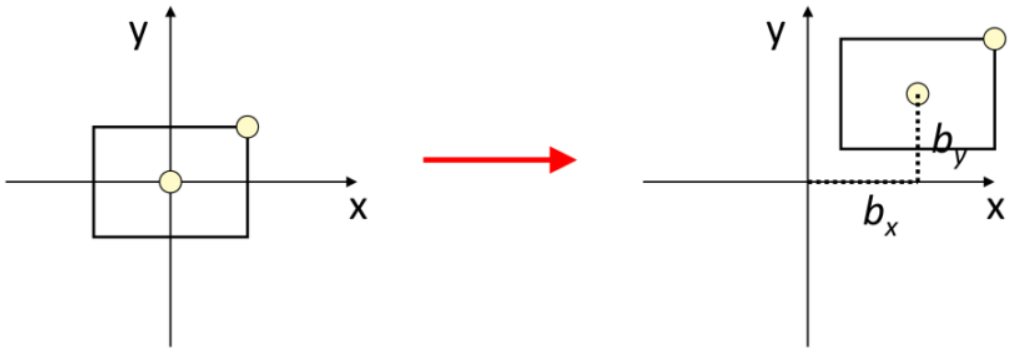
ПЕРЕНОС ИЗОБРАЖЕНИЙ

Трансляция изображения или перенос — это прямолинейный сдвиг изображения из одного места в другое, поэтому сдвиг объекта

называется переносом. Приведенное ниже матричное уравнение используется для переноса изображения:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & b_x \\ 0 & 1 & b_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Значение b_x определяет, насколько изображение будет перемещено по оси x, а значение b_y определяет перемещение изображения по оси y:



Теперь, когда вы поняли перенос изображений, давайте взглянем на код Python. В OpenCV есть две встроенные функции для выполнения такого преобразований:

- `cv2.warpPerspective`, которая принимает в качестве входных данных матрицу преобразования (3×3).
- `cv2.warpAffine` принимает матрицу преобразования (2×3) в качестве входных данных.

Обе функции принимают три входных параметра:

- Входное изображение.
- Матрица трансформации.
- Кортеж высоты и ширины изображения.

В этом уроке будем использовать функцию `cv2.warpPerspective()`.

Приведенный ниже код считывает входное изображение (если вам важна точность, получите изображение из этого урока здесь и поместите его в текущий рабочий каталог), переносит и показывает:

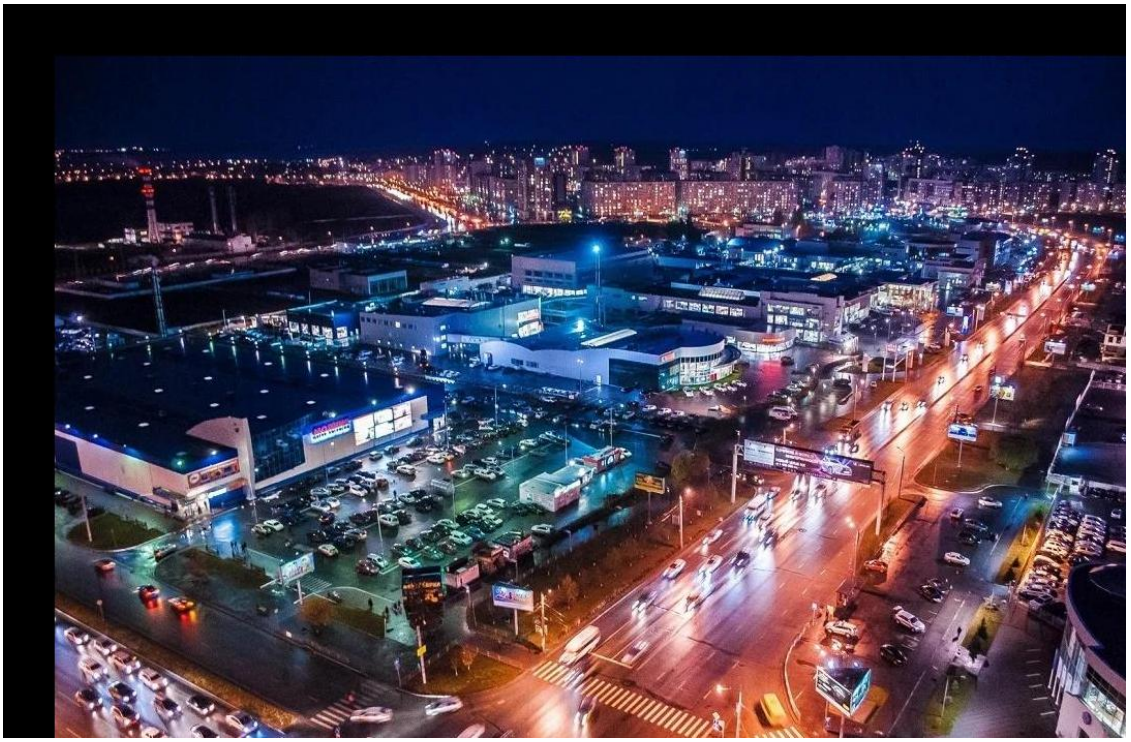
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

# читать входное изображение
img = cv2.imread("chelyabinsk.jpg")
# преобразовать из BGR в RGB, чтобы можно было построить график с помощью matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# отключить оси x и y
plt.axis('off')
# показать изображение
plt.imshow(img)
plt.show()
# получить форму изображения
rows, cols, dim = img.shape
# матрица преобразования для перевода
M = np.float32([ [1, 0, 50],
                 [0, 1, 50],
                 [0, 0, 1] ])
# применяем перспективное преобразование к изображению
translated_img = cv2.warpPerspective(img, M, (cols, rows))
# отключить оси x и y
plt.axis('off')
# показать получившееся изображение
plt.imshow(translated_img)
plt.show()
# сохраняем получившееся изображение на диск
plt.imsave("chelyabinsk_translated.jpg", translated_img)
```

Обратите внимание, что мы используем `plt.axis('off')`, поскольку мы не хотим выводить значения оси, и мы показываем изображение с помощью функции `imshow()` из **matplotlib**.

Также использована функцию `plt.imsave()` для локального сохранения изображения.

Исходное изображение:

Результат переноса:

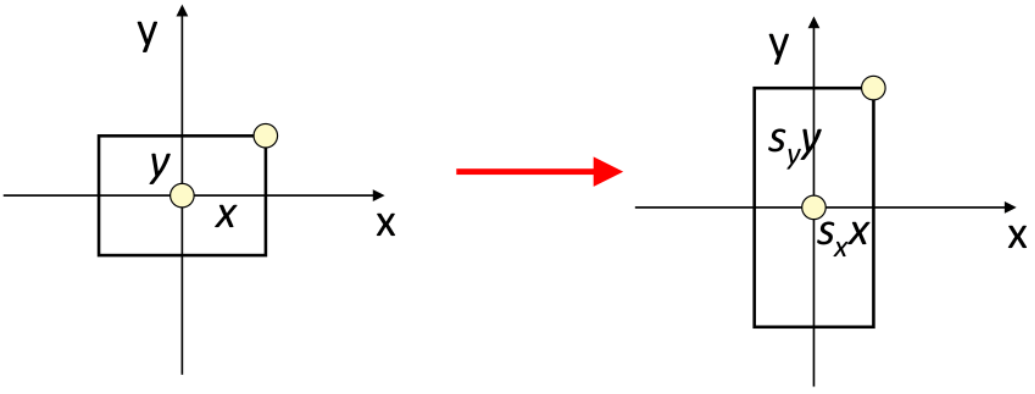


МАСШТАБИРОВАНИЕ ИЗОБРАЖЕНИЯ

Масштабирование изображения[1] — это процесс изменения размера цифрового изображения с сохранением пропорций. OpenCV имеет встроенную функцию `cv2.resize()`, но мы будем выполнять преобразование, используя умножение матриц, как и раньше. Матричное уравнение, используемое для масштабирования, показано ниже.

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

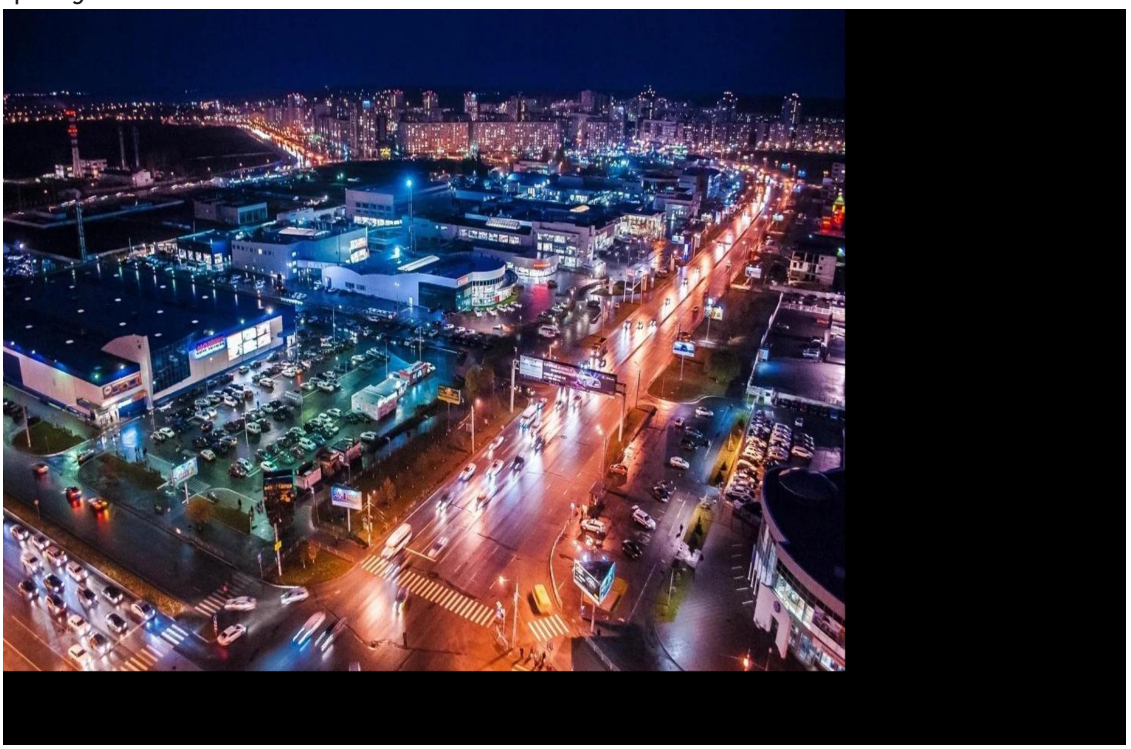
S_x и S_y — коэффициенты масштабирования для оси x и оси y соответственно.



Приведенный ниже код отвечает за чтение того же изображения, определение матрицы преобразования для масштабирования и показывает получившееся изображение:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

# читать входное изображение
img = cv2.imread("chelyabinsk.jpg")
# преобразовать из BGR в RGB, чтобы можно было построить график с помощью matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# отключить оси x и y
plt.axis('off')
# показать изображение
plt.imshow(img)
plt.show()
# получить форму изображения
rows, cols, dim = img.shape
# матрица преобразования для масштабирования
M = np.float32([ [1.5, 0  , 0],
                 [0,   1.8, 0],
                 [0,   0,   1] ])
# применяем перспективное преобразование к изображению
scaled_img = cv2.warpPerspective(img,M,(cols*2,rows*2))
# отключить оси x и y
plt.axis('off')
# показать получившееся изображение
plt.imshow(scaled_img)
plt.show()
# сохраняем получившееся изображение на диск
plt.imsave("chelyabinsk_scaled.jpg", scaled_img)
```

Вот результат:



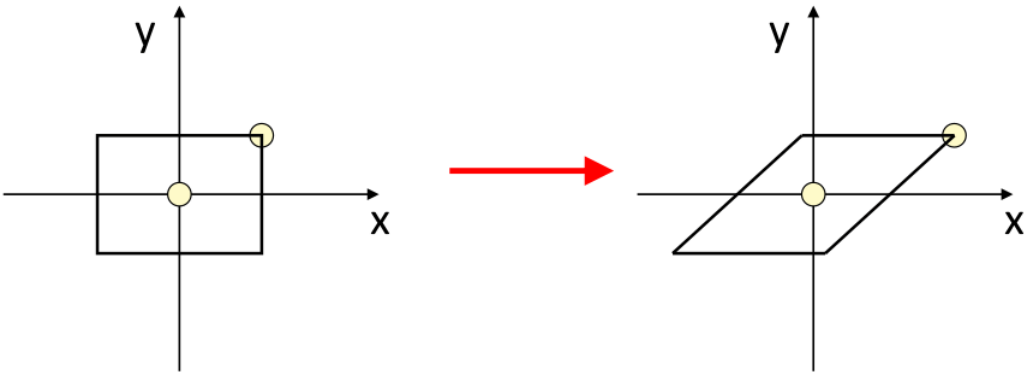
Обратите внимание, что вы можете легко удалить эти черные пиксели с помощью обрезки, но об этом в конце урока.

СКОЛЬЖЕНИЕ ИЗОБРАЖЕНИЯ

Карта сдвига — это линейная карта, которая смещает каждую точку в фиксированном направлении, она заменяет каждую точку по горизонтали или вертикали определенным значением, пропорциональным ее координатам x или y, есть два типа эффектов сдвига.

СКОЛЬЖЕНИЕ В НАПРАВЛЕНИЕ ОСИ X

Когда сдвиг выполняется в направлении оси x, границы изображения, параллельные оси x, сохраняют свое положение, а края, параллельные оси y, меняют свое положение в зависимости от коэффициента сдвига:



СКОЛЬЖЕНИЕ В НАПРАВЛЕНИЕ ОСИ Y

Когда сдвиг выполняется в направлении оси y, границы изображения, параллельные оси y, сохраняют свое положение, а края, параллельные оси x, меняют свое положение в зависимости от коэффициента сдвига.

Матричное уравнение для скольжения показано ниже:

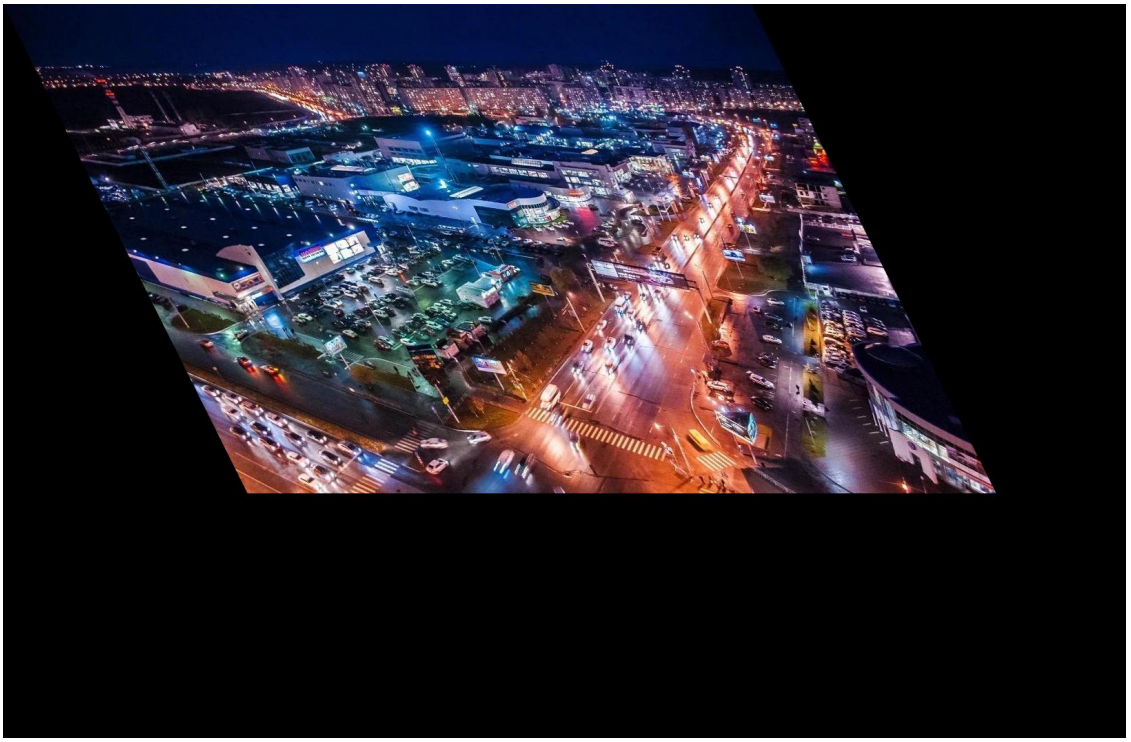
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} sh_x & 0 & 0 \\ 0 & sh_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Ниже приведен код для скольжения:

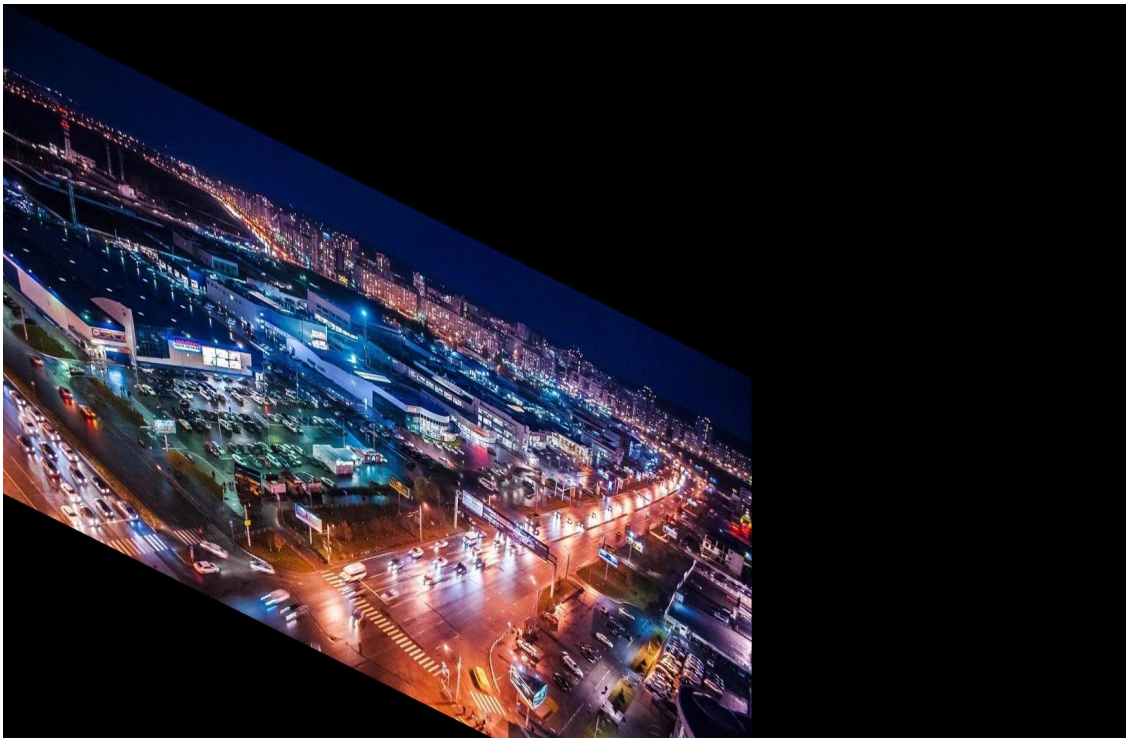
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

# читать входное изображение
img = cv2.imread("chelyabinsk.jpg")
# конвертировать из BGR в RGB, чтобы мы могли построить график с помощью matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# отключить оси x и y
plt.axis('off')
# показать изображение
plt.imshow(img)
plt.show()
# получить форму изображения
rows, cols, dim = img.shape
# матрицы преобразования для сдвига
# сдвиг, примененный к оси x
M = np.float32([ [1, 0.5, 0],
                 [0, 1  , 0],
                [0, 0  , 1] ])
# сдвиг, примененный к оси Y
# M = np.float32([ [1,   0, 0],
#                  [0.5, 1, 0],
#                  [0,   0, 1] ])
# применяем перспективное преобразование к изображению
sheared_img = cv2.warpPerspective(img,M,(int(cols*1.5),int(rows*1.5)))
# отключить оси x и y
plt.axis('off')
# show the resulting image
plt.imshow(sheared_img)
plt.show()
# сохраняем получившееся изображение на диск
plt.imsave("chelyabinsk_sheared.jpg", sheared_img)
```

Первая матрица — это сдвиг, примененный к оси x, если вы хотите ось y, то закомментируйте первую матрицу и раскомментируйте вторую.

Скольжение изображения по оси x:



Скольжение изображения по оси y:



ОТРАЖЕНИЕ ИЗОБРАЖЕНИЯ

Отражение изображения (или зеркальное отображение) переворачивает изображение как по вертикали, так и по горизонтали, это частный случай масштабирования. Для отражения по оси x мы устанавливаем значение S_y равным -1, а S_x равным 1 и наоборот для отражения по оси y.

Матричные уравнения преобразования для отражения показаны ниже:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & rows \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & cols \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Вот вам для размышлений код Python:

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

# read the input image
img = cv2.imread("chelyabinsk.jpg")
# convert from BGR to RGB so we can plot using matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# disable x & y axis
plt.axis('off')
# show the image
plt.imshow(img)
plt.show()
# get the image shape
rows, cols, dim = img.shape
# transformation matrix for x-axis reflection
M = np.float32([ [1,  0, 0   ],
                 [0, -1, rows],
                 [0,  0, 1   ] ])
# transformation matrix for y-axis reflection
# M = np.float32([ [-1, 0, cols],
#                  [ 0, 1, 0   ],
#                  [ 0, 0, 1   ] ])
# apply a perspective transformation to the image
reflected_img = cv2.warpPerspective(img,M,(int(cols),int(rows)))
# disable x & y axis
plt.axis('off')
# show the resulting image
plt.imshow(reflected_img)
plt.show()
# save the resulting image to disk
plt.imsave("chelyabinsk_reflected.jpg", reflected_img)
```

Как и раньше, сначала будет переворот относительно ось x (вертикальное отображение). Если хотите, раскомментируйте вторую матрицу и закомментируйте первую, получите переворот относительно оси y (вертикальное отображение).

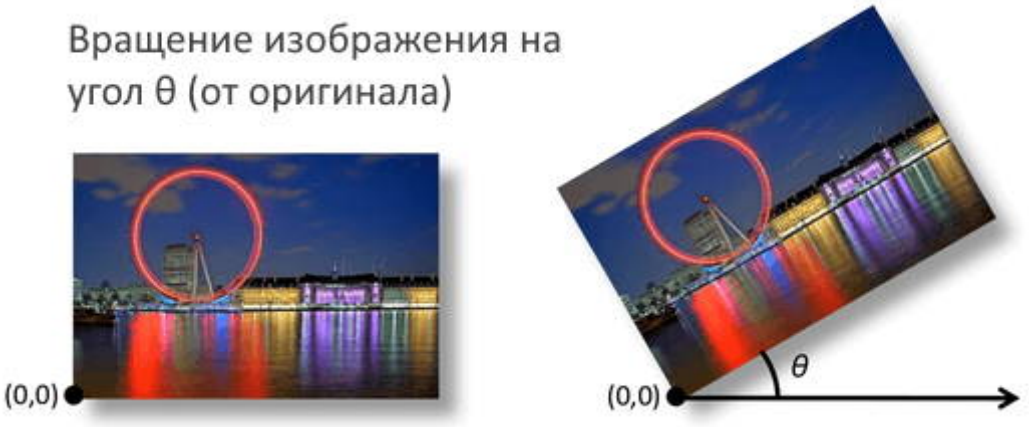
Переворот изображения относительно оси X:

Переворот изображения относительно оси Y:

ПОВОРОТ ИЗОБРАЖЕНИЯ

Вращение — это понятие в математике, которое представляет собой движение определенного пространства, которое сохраняет хотя бы одну точку. Поворот изображения — это обычная процедура обработки изображений с приложениями для сопоставления, выравнивания и других алгоритмов на основе изображений, а также

для увеличения данных, особенно когда речь идет о классификации изображений.



Матрица преобразования вращения показана на рисунке ниже, где θ — угол поворота:

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

Ниже приведен код Python для поворота изображения:

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

# читать входное изображение
img = cv2.imread("chelyabinsk.jpg")
# преобразовать из BGR в RGB, чтобы можно было построить график с помощью matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# отключить оси x и y
plt.axis('off')
# показать изображение
plt.imshow(img)
plt.show()
# получить форму изображения
rows, cols, dim = img.shape
# угол от градуса до радиана
angle = np.radians(10)
# матрица преобразования для вращения
M = np.float32([ [np.cos(angle), -(np.sin(angle)), 0],
                 [np.sin(angle), np.cos(angle), 0],
                 [0, 0, 1] ])
# применяем перспективное преобразование к изображению
rotated_img = cv2.warpPerspective(img, M, (int(cols),int(rows)))
# отключить оси x и y
plt.axis('off')
# показать получившееся изображение
plt.imshow(rotated_img)
plt.show()
# сохраняем получившееся изображение на диск
plt.imsave("chelyabinsk_rotated.jpg", rotated_img)
```

Вот и повернули изображение:

Оно было повернуто на 10° `(np.radians(10))`, вы можете редактировать код по своему усмотрению!

ОБРЕЗКА ИЗОБРАЖЕНИЯ

Обрезка изображения — это удаление нежелательных внешних областей с изображения, многие из приведенных выше примеров включают черные пиксели, вы можете легко удалить их с помощью обрезки. Код ниже делает это:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

# читать входное изображение
img = cv2.imread("chelyabinsk.jpg")
# преобразовать из BGR в RGB, чтобы можно было построить график с помощью matplotlib
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# отключить оси x и y
plt.axis('off')
# показать изображение
plt.imshow(img)
plt.show()
# получаем 200 пикселей от 100 до 300 по оси x и оси y
# измените это, если хотите, просто убедитесь, что вы не превышаете столбцы и строки
cropped_img = img[100:300, 100:300]
# отключить оси x и y
```

```
plt.axis('off')
# показать получившееся изображение
plt.imshow(cropped_img)
plt.show()
# сохраняем получившееся изображение на диск
plt.imsave("chelyabinsk_cropped.jpg", cropped_img)
```

Поскольку OpenCV загружает изображение в виде массива **numpy**, то можно обрезать изображение, просто указав нужные индексы нарезки, в нашем случае мы решили получить изображение размером **200x200** пикселей, вырезанное из исходного с 100 по 300 пиксели по обеим осям. Вот результат:

ЗАКЛЮЧЕНИЕ

В этом уроке мы рассмотрели основы обработки и преобразования изображений, которые включают перенос изображения, масштабирование, сдвиг, отражение, поворот и обрезку.

Вы можете получить все коды здесь.

Основа Image Transformations using OpenCV in Python

Респект и уважуха

К ВВ / 23/05/2021 / Цифровая трансформация / opencv, изображение, трансформация