TP5_Geometric_Transformations

December 1, 2023

#

Imagerie Numérique 2023 Automne

December 1st, 2023

#

TP Class N°5 - Geometric Transformations

Instructions :

- This TP should be completed and uploaded on Moodle before **Thursday 7 December 2023, 23h59**.
- The name of the file you upload should be **TP5_name_surname.ipynb**.
- If you need to include attached files to you TP, please archive them together in a folder named **TP5_name_surname.zip**.

General Advice:

When doing image processing and performing operations on pixels, it is a good practice to use float values for pixels with intensity [0,1]. Performing operations on uint8 encoded images can result in integer overflow and thus compute unpredictable values.

0.1 Exercise 1 : Image rotations

(2 points)

Import the RGB image : "mushroom.jpg".

(a) Using the geometric function *rotate* in the package *skimage.transform*, apply a 30° anti-clockwise rotation to the image and a 100° clockwise rotation. Visualize all three images side-by-side.

[ ]:

(b) Explain the meaning of the parameters 'center' and 'resize' of the function. Comment the effects on the border. Explain it on your images.

[ ]:

(c) Starting with the original image :

- Apply a rotation of 10 ° to the image and repeat this operation 36 times (this will perform a full 360 ° rotation). Try different parameters 'order' (=0,1,2,3). Visualize the original image and the results side-by-side.
- Compute MSE between the original image and the various results you obtained. **Hint :** Pay attention to the pixels you apply your MSE measure to, you may want to use a mask.
- What is the effect of the parameter 'order' ? How does it work ?

[ ]:

0.2 Exercise 2 : QR code reading

(2 points)

In this exercise, you will implement a simple QR code reader based on the image *QR_code_persp.jpg*

(a) Start by loading the image and convert it to grayscale. Visualize it. Locate the four corners of the QR code in pixel coordinates.

[ ]:

(b) You now want to apply a projective transform to your QR code to have it squared and well-aligned. To do this, you will use the class *ProjectiveTransform()* in the package *skimage.transform* (see an example here).

- Use the method *estimate()* with a source shape $[[0,0],[0,610],[610,610],[610,0]]$
- Visualize the projective matrix which is stored in the params attribute of your ProjectiveTransform() object.
- Explain the meaning of the coefficients of this matrix.

[ ]:

(c) Apply the projective transform to your image

- Use the function *warp()* from *skimage.transform* with you ProjectiveTransform object as argument.
- Visualize the transformed image.

[ ]:

(d) What further steps would you consider in order to program a QR code reader ?

– your answer –

0.3 Exercise 3 : Nearest interpolation

(2 points)

In this exercise, you will write a program that performs rescaling of images using nearest interpolation.

(a) Load the image 'lena.png' and convert it to grayscale. Perform a downsampling, taking one pixel every 3 horizontally and vertically.

[ ]:

(b) Write a function that performs resizing using the nearest interpolation and apply it to the downsampled image.

[63]:
```
def resize_ni(img, output_shape):
    """
    Changes the shape of the input image to match the output_shape

    Parameters
    ----------
    img : numpy array
        The input grayscale image

    output_shape : numpy array
        A couple of integers (x,y) giving the shape of the output image.

    Outputs
    ------
    resized : numpy array
        The image after the resizing algorithm. It should have shape␣
 ↪output_shape.
    """

    return resized
```

Hint : You may want to use *np.round()* to find the nearest-neighbouring point. Try not to use loops on pixels as this is veeeery slow.

(c) Apply the function *resize()* from *skimage.transform* with parameter 'order'=0. Visualize all three images (original, resize_ni and resize) side-by-side. Compute MSE between the original and the upscaled images. Comment your results.

[ ]: