Computer Vision Transformation

Tassadaq Hussain

Associate Prof: Riphah Int'l University
Microsoft Barcelona Supercomputing Center Barcelona, Spain
UCERD Pvt Ltd Islamabad

Image color spaces

RGB: Probably the most popular color space. It stands for Red, Green, and Blue. In this color space, each color is represented as a weighted combination of red, green, and blue. So every pixel value is represented as a tuple of three numbers corresponding to red, green, and blue. Each value ranges between 0 and 255.

YUV: Even though RGB is good for many purposes, it tends to be very limited for many real-life applications. People started thinking about different methods to separate the intensity information from the color information. Hence, they came up with the YUV color space. Y refers to the luminance or intensity, and U/V channels represent color information. This works well in many applications because the human visual system perceives intensity information very differently from color information.

HSV: As it turned out, even YUV was still not good enough for some applications. So people started thinking about how humans perceive color, and they came up with the HSV color space. HSV stands for Hue, Saturation, and Value. This is a cylindrical system where we separate three of the most primary properties of colors and represent them using different channels. This is closely related to how the human visual system understands color. This gives us a lot of flexibility as to how we can handle images.

Converting Color Space

```
import cv2
img = cv2.imread('./images/input.jpg', cv2.IMREAD_COLOR)
gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
cv2.imshow('Grayscale image', gray_img)
cv2.waitKey()
```

You can convert to YUV by using the following flag:

```
yuv_img = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)

y,u,v = cv2.split(yuv_img)
cv2.imshow('Y channel', y)
cv2.imshow('U channel', u)
cv2.imshow('V channel', v)
cv2.waitKey()
```

Image rotation

```
import cv2
import numpy as np
img = cv2.imread('images/input.jpg')
num_rows, num_cols = img.shape[:2]
rotation_matrix = cv2.getRotationMatrix2D((num_cols/2, num_rows/2), 30, 0.7)
img_rotation = cv2.warpAffine(img, rotation_matrix, (num_cols, num_rows))
cv2.imshow('Rotation', img_rotation)
cv2.waitKey()
```

Image scaling

```
import cv2
img = cv2.imread('images/input.jpg')
img_scaled = cv2.resize(img,None,fx=1.2, fy=1.2, interpolation = cv2.INTER_LINEAR)
cv2.imshow('Scaling - Linear Interpolation', img_scaled)
img_scaled = cv2.resize(img,None, fx=1.2, fy=1.2, interpolation = cv2.INTER_CUBIC)
cv2.imshow('Scaling - Cubic Interpolation', img_scaled)
img_scaled = cv2.resize(img,(450, 400), interpolation = cv2.INTER_AREA)
cv2.imshow('Scaling - Skewed Size', img_scaled)
cv2.waitKey()
```

Transformations

Euclidean transformations are a type of geometric transformation that preserve length and angle measures.

If we take a geometric shape and apply Euclidean transformation to it, the shape will remain unchanged. It might look rotated, shifted, and so on, but the basic structure will not change. So technically, lines will remain lines, planes will remain planes, squares will remain squares, and circles will remain circles.

In Affine transformations, lines will remain lines, but squares might become rectangles or parallelograms. Basically, affine transformations don't preserve lengths and angles.

Scale-invariant feature transform (SIFT)

SIFT builds a pyramid by downsampling an image and taking the difference of Gaussian.

```
import cv2
import numpy as np
input_image = cv2.imread('images/fishing_house.jpg')
gray_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
# For version opencv < 3.0.0, use cv2.SIFT()
sift = cv2.xfeatures2d.SIFT_create()
keypoints = sift.detect(gray_image, None)
cv2.drawKeypoints(input_image, keypoints, input_image, \
flags = cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('SIFT features', input_image)
cv2.waitKey()
```

UCERD
Gathering Intellectuals
www.ucerd.com

BSC~Microsoft Research Centre