PROJE

Haftalık Rapor – 02.05.2021

2 MAYIS 2021

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

İÇİNDEKİLER

ÖZET

Görüntü damgalama daha derinlemesine işlenmiş ve uygulanmıştır. Yapılan damgalama saydamlık, alfa, beta vb. parametreler kullanılarak, bitwise işlemler ile, RGB değerlerin değişimi ile gerçekleştirilmiştir. Bu sayede görüntü damgasının silinmesi de mümkün kılınmıştır. Bu damganın silinmesi de damganın uygulanması gibi uygulamalı olarak anlatılmıştır.

GERİ DÖNÜŞLÜ GÖRÜNTÜ DAMGALAMA UYGULAMAK

Python'da sıkça kullanılan ve artık resmi kütüphanelerden sayılan OpenCV[1] kütüphanesi ile geri dönüşlü bir damgalama uygulması yapılmaya çalışılmıştır. Bu çerçevede kütüphanelerin yüklenmesi/çalıştırılması ve meydana gelen hataların giderilmesi bu rapor özelinde paylaşılmayacaktır. Gerçekleşen hataların düzeltilmesi, gereksinimlerin kodlanması ve ortamların stabilizasyonu ciddi süreler almıştır. Rapor özelinde paylaşılan komutlarda –satır araları- esinlenilmiştir.[2] Temel mantığı yüzde yüz kavranmış ve uygulaması gerçek zamanlı yapılmıştır. Aynı zamanda tüm bu süreç sancılı geçeceği bilindiği üzere daha fazla hakim olunan PyCharm Community Edition[3] kullanılmıştır. Ayrıca step-step görsellerin rapora yansıması literatürde bulunan makalelerin hiçbirinde görünen bir durum değildir. Bu sebeple oluşturulan örnek görseller üzerinden yapılan işlemlerin gösterilmesi telif hakları korunumu için gösterilmemesine özen gösterilmiştir. Aitlik barındıran görseller kullanılamadığı için (önceki raporda; Kırıkkale Üniversitesi logosu bile izinsiz bir şekilde kullanılmıştır.) her defasında gösterilmesi doğru bulunmamıştır. Bir de yalnızca durağan görseller[4] değil hareketli görseller[5] de OpenCV ile birlikte uygulamalarda kapsam içerisine dahil edilmiştir. Bu sebeple dosya yollarında dosyaların bulunduğu varsayılarak aşağıdaki kodlamalar düzenlenmiştir.

PyCharm 1

```
from imutils import paths
import cv2
import numpy as np
import argparse
import os
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--watermark", required = True, help = "path
to watermark image (assumed to be transparent PNG)")
ap.add_argument("-i", "--input", required = True, help = "path to
the input directory of images")
ap.add_argument("-o", "--output", required = True, help = "path to
the output directory")
ap.add_argument("-a", "--alpha", type = float, default = 0.25,
```

[1] Resmi OpenCV web sayfası Referans ve Kaynakça bölümünde paylaşılmıştır.
[2] Esinlenilen web sayfası Referans ve Kaynakça bölümünde paylaşılmıştır.
[3] Kullanılan sürüm ve detayları için tıklayınız.
[4] Durağan görselden kasıt; .jpg, .jpeg, .png uzantılı dosyalardır.
[5] Hareketli görselden kasıt; .mp4, .gif uzantılı dosyalardır.

```
help = "alpha transparency of the overlay (smaller is more
transparent)")
ap.add_argument("-c", "--correct", type = int, default = 1, help =
"flag used to handle if bug is displayed or not")
args = vars(ap.parse_args())
alpha = args["alpha"]
beta = 1-args["alpha"]
print("alpha" + str(alpha))
print("beta" + str(beta))
watermark = cv2.imread(args["watermark"], cv2.IMREAD_UNCHANGED)
(wH, wW) = watermark.shape[:2]
if args["correct"] > 0:
    (B, G, R, A) = cv2.split(watermark)
    B = cv2.bitwise_and(B, B, mask = A)
    G = cv2.bitwise_and(G, G, mask = A)
    R = cv2.bitwise_and(R, R, mask = A)
    watermark = cv2.merge([B, G, R, A])
for imagePath in paths.list_images(args["input"]):
    image = cv2.imread(imagePath)
    (h, w) = image.shape[:2]
    image = np.dstack([image, np.ones((h, w), dtype = "uint8") *
255])
    overlay = np.zeros((h, w, 4), dtype = "uint8")
    overlay[h - wH - 10:h - 10, w - wW - 10:w - 10] = watermark
    output = image.copy()
    cv2.addWeighted(overlay, alpha, output, beta, 0, output)
    filename = imagePath[imagePath.rfind(os.path.sep) + 1:]
    p = os.path.sep.join((args["output"], filename))
    cv2.imwrite(p, output)
```

PyCharm 1'de sırasıyla yapılanlar anlatılacak olursa:

1. Run time parametreleri tanımlanmıştır. Damgalama, girdi ve çıktı bilgileri, opaklık gibi parametreler ismen tanımlanıp çalışma zamanında özel isimler ile değiştirilecektir.
2. RGB tonlar bitwise işlemler ile maskelenmiştir.(AND'lenmiştir.)
3. Girdi bilgisine uygun çıktı damgalanarak oluşturulmuştur.

Bu adımlar sayesinde aslında daha evvel yapılan ve pillow kütüphanesi kullanılarak gerçekleştirilen damgalama neredeyse bire bir şekilde uygulanmıştır. Bu iki yöntemin farkları aslında şudur:

1. Pillow opaklığın değiştirilmesine kısmen izin verse de OpenCV bitwise işlemlerini etkin bir şekilde kullandığı için ve RGB olarak tonları ayrı ayrı muhafaza ettiği için opaklık üzerindeki değişiklikler daha esnektir.

2. Open Source mantığı ile paylaşılan komutlar özel içerikler (örnek görsel adı, girdi dosya yolu, çıktı dosya yolu vb.) barındırmaz. Bu özel içerikler parametre olarak çalışma zamanında atanabilir. Bu sayede telif hakkı vb. korumalar daha etkin bir şekilde kullanılmış olur.
3. Bitwise olarak işlemler yapıldığı için yapılan damgalama geri dönüşümlü (feedbacking) olarak tasarlanabilir. Marking - Demarking işlemleri yapılabilmesine imkan sağlar.

GÖRÜNTÜ DAMGASININ TEMİZLENMESİ

Bir önceki konu itibariyle gerçekleştirilen ve PyCharm 1'de komutların paylaşıldığı damgalama yöntemine birkaç değişiklik yaparak[6] damganın temizlenmesi sağlanacaktır. Bu temizlik demarking yerine remove marking olarak da literatürde geçmektedir. İki ifade arasında aslında teorik anlamda bir fark mevcuttur fakat pratikte yapılan iş aynıdır. Pratikte bahsi geçen işlem şudur: Daha evvel damgalanmış bir görüntünün damgasından temizlenmesidir.

Temel mantık tamamen katmanlar arası geçişle bağıntılıdır. Temizlik tamamen alfa ve beta değerlerinin değiştirilmesi ile elde edilmiştir.

PyCharm 2

```
from imutils import paths
import numpy as np
import argparse
import cv2
import os
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--watermark", required = True, help = "path
to watermark image (assumed to be transparent PNG)")
ap.add_argument("-i", "--input", required = True, help = "path to
the input directory of images")
ap.add_argument("-o", "--output", required = True, help = "path to
the output directory")
ap.add_argument("-a", "--alpha", type = float, default = 0.25,
help = "alpha transparency of the overlay (smaller is more
transparent)")
ap.add_argument("-c", "--correct", type = int, default = 1, help =
"flag used to handle if bug is displayed or not")
args = vars(ap.parse_args())
alpha = 1 / (1 - args["alpha"])
beta = (1 - 1 / (1 - args["alpha"]))
print("alpha" + str(alpha))
print("beta" + str(beta))
watermark = cv2.imread(args["watermark"], cv2.IMREAD_UNCHANGED)
(wH, wW) = watermark.shape[:2]
if args["correct"] > 0:
    (B, G, R, A) = cv2.split(watermark)
    B = cv2.bitwise_and(B, B, mask = A)
    G = cv2.bitwise_and(G, G, mask = A)
    R = cv2.bitwise_and(R, R, mask = A)
    watermark = cv2.merge([B, G, R, A])
for imagePath in paths.list_images(args["input"]):
    image = cv2.imread(imagePath)
    (h, w) = image.shape[:2]
    image = np.dstack([image, np.ones((h, w), dtype="uint8") *
255])
```

[6] PyCharm 1'deki parametre ve komutlarda yapılacak değişiklik kastedilmiştir.

```
    overlay = np.zeros((h, w, 4), dtype="uint8")
    overlay[h - wH - 10:h - 10, w - wW - 10:w - 10] = watermark
    output = image.copy()
    cv2.addWeighted(output, alpha, overlay, beta, 0, output)
    filename = imagePath[imagePath.rfind(os.path.sep) + 1:]
    p = os.path.sep.join((args["output"], filename))
    cv2.imwrite(p, output)
```

PyCharm 2'de output daha evvel söylendiği gibi overlay ile değiştirilmiştir. Parametreler de buna oranla değişmiştir. PyCharm 2'de alfa ve beta değerleri tersine çevrilebilir düzeyde tutulmuştur. Bu da reverse (görüntünün ters çevrimi) için kullanılmıştır. Bu sayede PyCharm 1'de yapılan damgalama PyCharm 2'deki birkaç değişiklik ve aslında göz yanılsaması ile temizlenmiştir.

ANAHTAR KELİMELER

- ***PyCharm X*** : X. PyCharm Community Edition çıktısı.

REFERANS VE KAYNAKÇA

- OpenCV Kütüphanesi – Link için tıklayınız.
- Numpy Kütüphanesi – Link için tıklayınız.
- PyCharm Community Edition – Link için tıklayınız.
- Numpy görüntü işlemesi üzerine bir yazı – Link için tıklayınız.
- Methodologies in Digital Watermarking: Robust and Reversible Watermarking Techniques for Authentication, Security and Privacy Protection by Xin Cindy Guo – Link için tıklayınız.
- Dijital Damgalama: Öğretici, Dr. Vipula Singh – Link için tıklayınız.