Универзитет „Св. Кирил и Методиј“– Скопје

Факултет за информатички науки и компјутерско инженерство (ФИНКИ)

Семинарски труд по предметот: Дигитално процесирање на слика

ТЕМА: **Cartoonifying an Image**

Ментор:
проф. д-р Ивица Димитровски

Кандидати:
Дарко Васиљков 213030

Христина Стојановска 211031

Куманово, јуни 2023.

Содржина

Куманово, јуни 2023.

Вовед

Во современото време, влијанието на комиксите и анимациите се забележува во голема мера во популарната култура. Од своите светли бои, уникатни стилови до пренесувањето на емоции, стриповите и анимациите привлекуваат внимание. Во последно време, постои пораст на интересот за трансформација на фотографии и слики во стил на стрип, со цел да добијат уникатен и забавен изглед.

Ова воведување ќе го истражи процесот на претворање на слики во стил на стрип со помош на библиотеката OpenCV и програмскиот јазик Python. OpenCV (Open Source Computer Vision) е популарна библиотека за компјутерска визија која овозможува обработка на слики и видеа. Со помош на OpenCV и Python, можеме да карикатуризираме слики и да ги трансформираме во стил што приличи на стрип.

Процесот на карикатуризација вклучува неколку чекори. Прво, сликата се претвора во црно-бела верзија за да се намали комплексноста на анализата. Потоа, се применуваат техники за детекција на контури, со што се откриваат главните облини на сликата. Следно, се додаваат линии и нијанси за да се постигне стрип ефектот. На крај, се додаваат светли бои и шари, што го зголемува комикс визуелниот стил на сликата.

Имплементирањето на овој процес се постигнува со помош на Python и OpenCV. Python е популарен програмски јазик во областа на машинско учење и компјутерска визија, што го прави идеален за работа со OpenCV. OpenCV ни овозможува да применуваме различни филтри, алгоритми и техники за обработка на слики, што е критично за постигнување на карикатурскиот ефект

Во продолжение на ова воведување, ќе ги истражиме различните техники и алгоритми на располагање во OpenCV и Python за карикатуризирање на слики. Ќе ги разгледаме различните пристапи и методи што се користат во овој процес, и како можеме да ги примениме за да ги трансформираме нашите слики во стрип стил.

Со претставување на процесот на карикатуризација на слики со помош на OpenCV и Python, ова воведување ни дава основа за истражување и креирање на сопствени карикатурски креации. Креирањето на стрип стил на сликите може да биде забавен начин за изразување на креативноста и креирање на уникатни визуелни искуства.

Куманово, јуни 2023.

Инструкции за користење на OpenCV

OpenCV (Open Source Computer Vision) е популарна отворена библиотека за компјутерска визија и обработка на слики. Таа нуди широк спектар на функции и алатки кои овозможуваат на програмерите да манипулираат и анализираат слики и видеа. Во ова ръководство, ќе истражиме основните чекори и концепти за користење на OpenCV во вашите проекти.

1. Инсталација:
   За да користите OpenCV, потребно е да го инсталирате на вашиот систем. Можете да го направите преку pip, Python пакетниот менаџер, со извршување на командата: pip install opencv-python.

2. Внесување на OpenCV:
   Штом го инсталирате, можете да го внесете OpenCV библиотеката во вашиот Python код со следнава наредба: import cv2. Тоа ви овозможува пристап до сите функции и класи кои ги нуди оваа библиотека.

3. Вчитување и Прикажување на Слики:
   OpenCV ви овозможува да вчитувате и манипулирате со слики во различни формати. За да вчитате слика, можете да користите функцијата cv2.imread(), која го прима патеката до сликата како параметар и враќа NumPy низа која ја претставува сликата. Потоа, можете да ја прикажете сликата со функцијата cv2.imshow().

4. Манипулација на Слики:
   OpenCV нуди голем број функции за манипулација на слики. Можете да извршите операции како промена на големината, одсечка, ротација и превртување на слики користејќи ги соодветните функции. Овие функции често бараат влезни параметри како низа на слика и враќаат изменета низа на слика.

5. Филтрирање и Обработка на Слики:
   OpenCV нуди низа од функции за филтрирање и обработка на слики. Можете да примените различни филтри, како Гаусовско заматување, медиумско заматување и детекција на ивици, користејќи ги функциите cv2.GaussianBlur(), cv2.medianBlur() и cv2.Canny(). Овие функции ви овозможуваат да ги засилите или извлечете специфични карактеристики од сликите.

6. Цртање на Слики:

OpenCV ви овозможува да цртате различни облици, линии и текст врз слики. Можете да користите функции како cv2.line(), cv2.rectangle(), cv2.circle() и cv2.putText() за цртање на облици и текст. Овие функции обично бараат влезни параметри како низа на слика, координати, боја и дебелина.

7. Детекција и Распознавање на Објекти:
   OpenCV вклучува предвчитани модели и алгоритми за детекција и распознавање на објекти. Можете да ги искористите овие модели, како Haar каскади или модели базирани на длабоко учење, за да детектирате и распознаете објекти на слики или видео потоци.

8. Управување со Ресурси:
   Важно е да ги ослободите ресурсите правилно откако ќе завршите со користење на OpenCV. На пример, користете cv2.destroyAllWindows() за да затворите отворени прозорци и ослободете ресурси за снимање на видео користејќи cv2.VideoCapture.release().

   Користејќи ја снагата на OpenCV, можете да извршувате различни задачи за компјутерска визија, да манипулирате со слики и видеа и да создавате иновативни апликации во области како роботика, зголемена реалност, надзор и повеќе.

Python

Python е едноставен и јасен интерпретациски јазик за програмирање, кој го прави одличен избор за почетници и напредни програмери. Тој се користи во различни области како веб развој, научно изчислување, машинско учење, анализа на податоци и скриптинг. Предностите на Python вклучуваат широка заедница и голем избор на библиотеки и модули, што овозможува брза и ефикасна развојна работа.

Овој објектно-ориентиран јазик работи со објекти или инстанци на класи, што го олеснува повторното искористување на кодот, модуларноста и организацијата на програмите во делови. Python користи читлива синтакса со употреба на разбирливи идентатори и облици на кодот, што го прави лесно разбирлив и изменлив.

Како интерпретиран јазик, Python не бара компилација пред извршување, што овозможува брза итерација и директна интеракција со кодот, што е посебно корисно при развој и експериментирање. Има богата стандардна библиотека и голем избор на третманси модули, што ги проширува неговите можности и го прави популарен во различни области.

Во целосност, Python е моќен алатка за програмирање со голема флексибилност и широк спектар на можностите. Бидејќи е едноставен и лесен за употреба, Python претставува отворена врата во светот на програмирањето и овозможува создавање

на различни апликации и решенија.Во целосност, Python е силно алатка за програмирање со голема флексибилност и моќ, која може да биде искористена за различни цели. Бидејќи е едноставен и лесен за употреба, Python е отворена врата за влез во светот на програмирањето и создавање на различни апликации и решенија.

1. Структура и техничка имплементација на проектот

За да го имплементираме дадениот проект за претворање на обична слика во стрип слика ни беше потребен програмскиот јазик Python. А како главна библиотека користевме OpenCV која ни го овозможи ова. Како Editor користевме PyCharm кој ни овозможи воедно и виртуелна околина за да го претставиме резултатот. Го создадовме проектот низ повеќе точки низ кои ќе проследиме во наредните пасуси.

1.1 Детектирање и нагласување на рабови

За да ги создадеме стрип ефектите како прв чекор треба да ја разбереме разликата помеѓу дигитална слика и слика во вид на цртан (стрип). Ќе го земеме примерот од овие 2 слики:

На прв поглед можеме јасно да видиме две големи разлики и тоа:

- Првата разлика е во тоа што боите на цртаната слика се похомогени во споредба со нормалната слика.
- Втората разлика е забележлива во рабовите кои се многу поостри и поизразени во цртаната слика.

Откако ги разбравме разликите може да започнеме со самата имплементација. Најпрвин ќе ги симнеме потребните библиотеки а потоа ќе ги импортнеме во самиот PyCharm – тоа би изгледало вака:

```
import cv2
import numpy as np
from tkinter.filedialog import *

1 usage
```

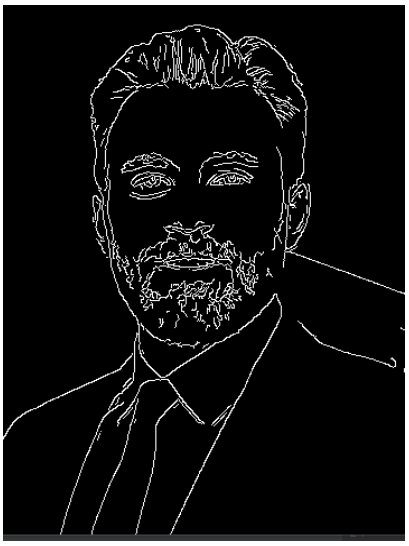
Потребно е потоа да ја вчитаме сликата, во овој случај ние ја вчитавме со помош на *tkinter.filedialog*. Поточно при секој Run на програмата ќе бидеме прашани за селекција на слика на која сакаме да и го извршиме исцртувањето.

```
photo = askopenfilename()
img = cv2.imread(photo)
cv2.imshow("Input Image", img)
cv2.waitKey(0)
```

Следниот чекор е откривање на рабовите. За таа задача треба да го избереме најпогодниот метод. Постојат неколку рабни детектори што можеме да ги избереме. Нашиот прв избор ќе биде еден од најчестите детектори, а тоа е детекторот ***Canny edge*.** Но, за жал, ако го примениме овој детектор нема да можеме да постигнеме посакувани резултати. Можеме да продолжиме со Кани, а сепак можете да видите дека има премногу детали снимени. Ова може да се смени ако си поигруваме со влезните параметри на Кани (Canny's) (т.е броеви 100 и 200).

```
edges = cv2.Canny(img, 100, 200)
cv2.imshow("Edges", edges)
cv2.waitKey(0)
```

Или тоа би изгледало вака на сликата:



Threshold image

0	0	0	0	255	0	0	0
255	0	0	0	0	0	0	0
0	0	0	0	0	0	255	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	255
0	0	0	0	0	255	0	0
0	255	0	0	0	0	0	0
0	0	255	0	0	0	0	0

Иако *Canny* е одличен детектор за рабови што можеме да го користиме во многу случаи во нашиот код, ќе користиме метод на праг што ни дава позадоволувачки резултати. Таа користи праг на пикселна вредност за да конвертира слика со сива скала во бинарна слика. На пример, ако вредноста на пикселот во оригиналната слика е над прагот, ќе биде доделена на 255. Во спротивно, ќе биде доделена на 0 како што можеме да видиме на следната слика.

Се одлучуваме да користиме функција ***cv2.adaptiveThreshold()*** која го пресметува прагот за помали региони на сликата. На овој начин, добиваме различни прагови за различни региони на иста слика. Тоа е причината зошто оваа функција е многу погодна за нашата цел. Ќе ги нагласи црните рабови околу предметите на сликата.

Значи, првото нешто што треба да направиме е да ја претвориме оригиналната слика во боја во слика со сива скала. Исто така, пред прагот, сакаме да го потиснеме шумот од сликата за да го намалиме бројот на откриени рабови кои се непожелни. За да го постигнеме ова, ќе го примениме медијалниот филтер кој ја заменува секоја вредност на пиксели со средната вредност на сите пиксели во мало соседство на пиксели. Функцијата cv2.medianBlur() бара само два аргументи: сликата на која ќе го примениме филтерот и големината на филтерот.

Како параметри користиме:

- максимална вредност која ќе биде поставена на 255
- **cv2.ADAPTIVE_THRESH_MEAN_C**: праг вредност е средната вредност на областа на соседството.
- **cv2.ADAPTIVE_THRESH_GAUSSIAN_C**: вредност на прагот е пондерираната сума на вредностите на соседството каде тежините се гаусовиот прозорец.
- **Големина на блок** - ја одредува големината на областа на соседството.
- **C** – Тоа е само константа која се одзема од пресметаната средина (или пондерираната средина).

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_1 = cv2.medianBlur(gray, 5)
edges = cv2.adaptiveThreshold(gray_1, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 5)
cv2.imshow("Edges with Threshold", edges)
cv2.waitKey(0)
```

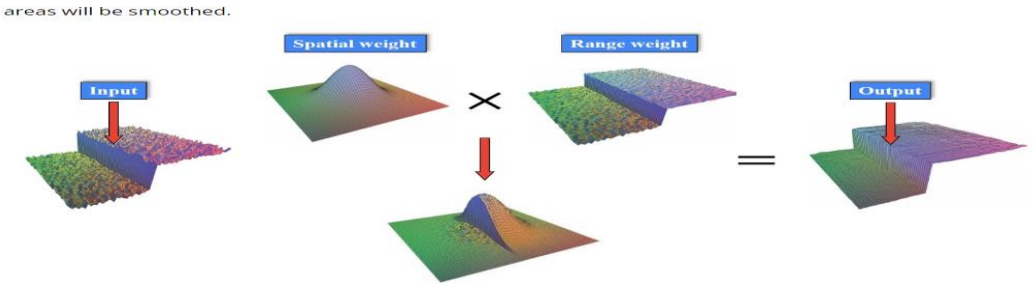
2. Филтрација на слика

Сега треба да избереме филтер кој е погоден за претворање на RGB слика во слика во боја или цртан филм. Постојат неколку филтри кои можеме да ги користиме. На пример, ако избереме да користиме филтер ***cv2.medianBlur()*** ќе добиеме солиден резултат. Ќе успееме да ги замаглиме боите на сликата за да изгледаат похомогени. Сепак ќе се одлучиме за Bilateral филтер бидејќи нуди работи кои баш ни требаат.

2.1 Билатерален филтер

Билатералниот филтер е еден од најчесто користените филтри за зачувување на рабовите и намалување на бучавата. Слично на Гаусовиот, билатералниот филтер ја заменува секоја вредност на пиксел со пондериран просек на вредностите на пиксели во близина. Сепак, разликата помеѓу овие два филтри е во тоа што билатералниот филтер ја зема предвид варијацијата на интензитетот на пикселите за да ги зачува рабовите. Идејата е дека два блиски пиксели кои зафаќаат блиски просторни локации, исто така, мора да имаат некоја сличност во нивоата на интензитет. Ова ќе го видиме со равенката:

Постојат три аргументи во функцијата ***cv2.bilateralFilter():***



Куманово, јуни 2023.

- **r** – Дијаметар на секое соседство на пиксели што се користи при филтрирање.
- **sigmaColor** – стандардна девијација на филтерот во просторот за боја. Поголема вредност на параметарот значи дека подалечните бои во соседството на пиксели ќе се мешаат заедно, што ќе резултира со поголеми површини со полуеднаква боја.
- **sigmaSpace** – стандардното отстапување на филтерот во координатниот простор. Поголема вредност на параметарот значи дека подалечните пиксели ќе влијаат еден на друг се додека нивните бои се доволно блиску.

```
color = cv2.bilateralFilter(img, d=9, sigmaColor=200, sigmaSpace=200)
cv2.imshow("Color Image", color)
cv2.waitKey(0)
```

3. Креирање на цртан филм ефект

Нашиот последен чекор е да ги комбинираме претходните две: ќе ја користиме функцијата ***cv2.bitwise_and()*** за мешање на рабовите и сликата во боја во една. Така ќе го добиеме конечниот резултат за тоа како би изгледала дигиталната слика во краен продукт слика на цртан филм (стрип).

```
cartoon = cv2.bitwise_and(color, color, mask=edges)
cv2.imshow("Cartoon", cartoon)
cv2.waitKey(0)
```

4. Креирање на стрип ефект користејќи боена квантизација.

Во прилог ќе го проследиме и процесот на друг начин на креирање на стрип ефект така што ќе го користиме методот на боена квантизација. За разлика од претходниот начин, овој начин дава по видлива слика на крајниот продукт па затоа избравме да го проследиме и него.

4.1 *Боена квантизација*

Ја користиме функцијата ***color_quantization()*** :

```
def color_quantization(img, k):
    data = np.float32(img).reshape((-1, 3))
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
    ret, label, center = cv2.kmeans(data, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
    center = np.uint8(center)
    result = center[label.flatten()]
    result = result.reshape(img.shape)
    return result
```

Различните вредности за \(K \) ќе го одредат бројот на бои на излезната слика. Значи, за нашата цел, ќе го намалиме бројот на бои на 7. Резултатот е даден во прилог:

```
img_1 = color_quantization(img, 7)
cv2.imshow("Color Quantization", img_1)
cv2.waitKey(0)
```

Сега е потребно пак да го искористиме Median филтерот на сликата за да го добиеме тоа хомогено замаглување:

```
blurred = cv2.medianBlur(img_1, 3)
cv2.imshow("Blurred Image", blurred)
cv2.waitKey(0)
```

Куманово, јуни 2023.

На крајот за целиот процес да е готов ќе ја споиме сликата со детекција на рабови со последната слика поточно хомогена замаглена слика која ја направиме во последниот чекор со постпаката на ***cv2.medianBlur(img_1, 3).***

Со оваа слика го добивме крајниот резултат кој е и целта на оваа семинарска. Добивме слика со ефект на цртан филм (стрип) низ повеќе чекори комбинирајќи различни филтери при што стигнавме до самиот резултат низ 2 начини и тоа првиот начин без боена квантизација, но и вториот кој е многу пореалистичен и кој ни го даде посакуваниот резултат на овој ефект.

Во прилог ќе бидат претставени слики заедно со крајниот ефект, за начинот на кој сме стигнале до истиот и прилози за секој чекор низ кој поминавме во оваа семинарска. За подобра споредба, ќе биде претставен и кодот како и секој output кој го добивме по стартување на кодот заедно со виртуелната средина.

Куманово, јуни 2023.

5. Комбинација на сите слики и краен резултат

5.1 Завршен код на програмата

```python
import cv2
import numpy as np
from tkinter.filedialog import *


1 usage
def color_quantization(img, k):
    data = np.float32(img).reshape((-1, 3))
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
    ret, label, center = cv2.kmeans(data, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)
    center = np.uint8(center)
    result = center[label.flatten()]
    result = result.reshape(img.shape)
    return result


photo = askopenfilename()
img = cv2.imread(photo)
cv2.imshow("Input Image", img)
cv2.waitKey(0)

edges = cv2.Canny(img, 100, 200)
cv2.imshow("Edges", edges)
cv2.waitKey(0)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray_1 = cv2.medianBlur(gray, 5)
edges = cv2.adaptiveThreshold(gray_1, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 5)
cv2.imshow("Edges with Threshold", edges)
cv2.waitKey(0)

color = cv2.bilateralFilter(img, d=9, sigmaColor=200, sigmaSpace=200)
cv2.imshow("Color Image", color)
cv2.waitKey(0)

cartoon = cv2.bitwise_and(color, color, mask=edges)
cv2.imshow("Cartoon", cartoon)
cv2.waitKey(0)
```

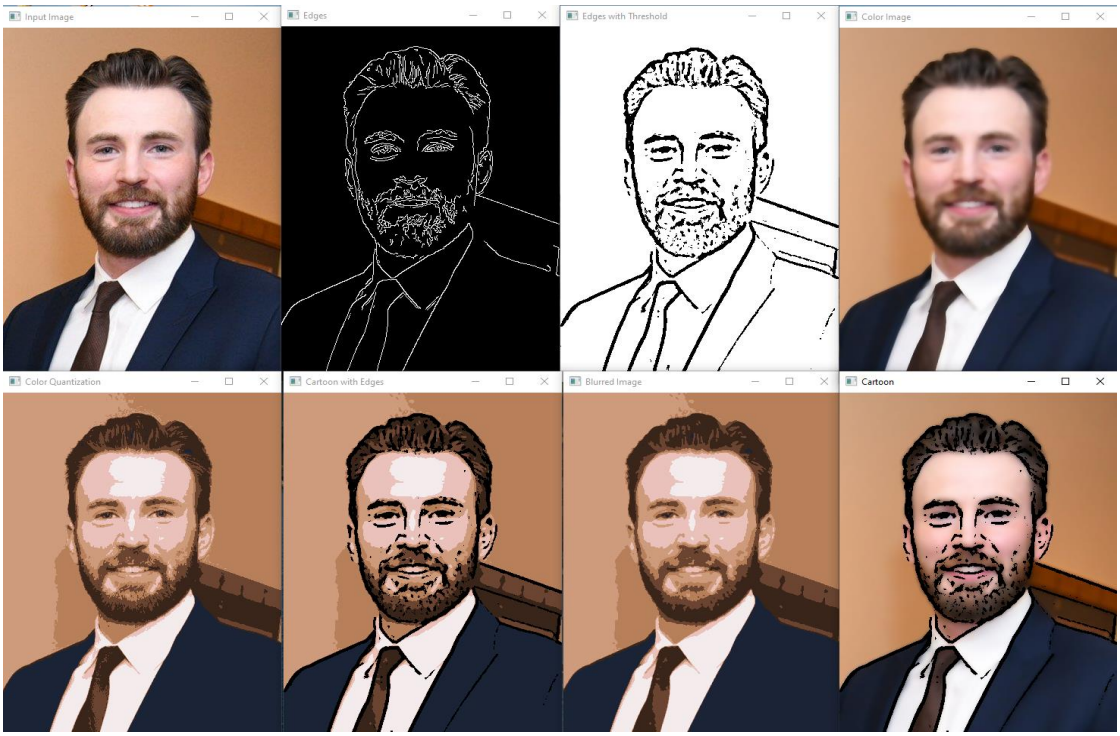
```python
img_1 = color_quantization(img, 7)
cv2.imshow("Color Quantization", img_1)
cv2.waitKey(0)

blurred = cv2.medianBlur(img_1, 3)
cv2.imshow("Blurred Image", blurred)
cv2.waitKey(0)

cartoon_1 = cv2.bitwise_and(blurred, blurred, mask=edges)
cv2.imshow("Cartoon with Edges", cartoon_1)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

5.2 Комбинација на сите слики заедно со крајниот резултат (цртан филм ефект)



Куманово, јуни 2023.

Заклучок

Преку оваа семинарска, направивме обработка на слики за да создадеме ефект на цртан филм (стрип). Овој процес вклучуваше наоѓање на ивици, претворање во сиво и адаптивно прагување за добивање на ивици со праг. Исто така, ги применувавме филтрите на билатерална филтрација и медијанска филтрација за да ги подобриме контурите и да го намалиме шумот во сликата. Дополнително, имплементиравме функција за квантизација на бои, која ги групира боите во кластери и ги заменува оригиналните пиксели со централните вредности. Ова ни помага да добиеме ефект на цртан филм со поедноставени бои.

На крај, ги прикажавме резултатите на сите чекори од процесот на ефектот на карикатура, вклучувајќи ги крајната слика, ивиците, ивиците со праг, сликата во боја, квантизацијата на бои, замаглената слика и цртан филм ефектот. Исто така, го прикажавме и дополнителниот резултат - карикатурата без квантизација на бои. Овој код ни овозможува да креираме ефектни цртан филм ефекти од сликите и да ги контролираме нивните аспекти, како што се контурите, боите и нивната едноставност.

Овој проект на обработка на слики за ефект на цртан филм ни овозможи да се забавуваме и креираме уникатни и креативни визуелни резултати преку употреба на различни техники и филтри кои ги научивме во текот на овој курс.

Референци

[1] OpenCV – Python tutorials - https://docs.opencv.org/4.x/

[2] Python tutorials - https://docs.python.org/3/

[3] Matplotlib.pyptlot.imshow - Documentation - https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html

[4] Prof. Dr. Vladimir Matic, Davor Jordacevic, Strahinja Zivkovic – “The Hundred-Page Computer Vision OpenCV Book”

[5] Bilateral filter - https://en.wikipedia.org/wiki/Bilateral_filter

Куманово, јуни 2023.