Факултет за информатички науки и компјутерско нженерство

ФИНКИ

ДИГИТАЛНО ПРОЦЕСИРАЊЕ НА СЛИКА

Тема:

Background Subtraction

Изработија:

Стефан Лазаревски 226079

Ана Марија Краус 226135

Ментор:

проф. д-р Ивица Димитровски

Содржина

Вовед

Одземањето на позадина е широко користена техника за обработка на слики за откривање на подвижни објекти во статични сцени или одвојување на преден план (foreground) од позадина (background) во неподвижни слики или видео. Вклучува изолирање и отстранување на позадината на сликата, така што примарниот објект (честопати наречен преден план) може да се обработи, анализира или користи одделно. Оваа техника најчесто се користи во апликации како што се:

- Компјутерски вид (на пр., откривање на објекти, следење)
- Видео надзор
- Проширена реалност (AR)
- Ефекти на зелен екран во видео продукција
- Уредување и компонирање на слики
- Медицинско снимање и научна визуелизација

Главната цел е да се подобри видливоста и употребливоста на примарниот објект на сликата со елиминирање на неважните визуелни информации (позадината). Одземањето на позадина е од суштинско значење кога:

- Потребно е чисто извлекување на објектот без рачно маскирање.
- Автоматизираните системи треба да детектираат движење или објекти во кадар.
- Создавање на множества податоци од слики каде што треба да остане само објектот од интерес.
- Подобрување на перформансите на моделите за машинско учење и длабинско учење со елиминирање на шумот од неважни пиксели.
- Поедноставување на интеракцијата со корисниците во креативни алатки, на пр., при креирање минијатури, слики на профили.

Нашата апликација е алатка за Desktop GUI изградена со Python и Tkinter што им овозможува на корисниците да вршат одземање на позадина на статички слики или видео во реално време користејќи повеќе методи. Апликацијата комбинира повеќе пристапи за одземање на позадина, секој прилагоден за различни случаи на употреба и преференции на корисниците. Инспирацијата зад развојот на оваа апликација произлегува од зголемената побарувачка за едноставни, лесни за користење алатки што можат да извршуваат сложени задачи за обработка на слики како што е отстранување на позадина без да бараат професионален софтвер како Photoshop. За таа цел нашата идеја беше да изградиме интуитивна, едукативна и функционална алатка, која овозможува избор на различни методи за одземање на позадина (автоматски, квадртатна селекција, селекција со четка и видео во реално време). Ова го овозживме со помош на OpenCV и GUI програмирање со Tkinter.

Преглед на концептот

Апликацијата нуди четири главни методи за одземање на позадината:

- **Автоматско одземање** – Користи метод за отстранување на надворешна позадина за автоматско сегментирање на објектот.
- **Одземање со избор на квадрат** – Корисникот избира квадратна област што го содржи објектот, а позадината надвор од тој квадрат се отстранува.
- **Одземање со избор на четка** – Корисникот рачно го означува предниот дел со помош на алатка со четка за прецизна контрола.
- **Одземање на видео позадина во реално време** – Ги снима влезните податоци од веб-камерата и се обидува да ја одземе позадината во реално време.

Секој метод може да се избере од паѓачко мени, а корисниците можат да ги прегледаат и оригиналната и обработената слика една до друга. Апликацијата, исто така, поддржува преземање на обработената слика за понатамошна употреба.

Исто така нуди и функционалност за **Додавање на позадина на одредена слика**, односно фунцкионалност за преклопување на транспарента предна слика (која може да се добие со користење на некој од претходно, горенаведените методи) со друга позадинска слика. Откако ќе се постават посакуваните слики, позадинска и предна и по избор на: „Replace background“ од паѓачкото мени се прикажува конечната споена слика. Исто така овозможува преземање на споениот резултат.

Се разбира, сегашната верзија на аплицаијата, односно нашата демо апликација, е мала, лесно проширлива платформа што подоцна може да се подобри со функции базирани на вештачка интелигенција или обработка во облак.

Технолошки компоненти

Нашата демо апликација е изградена со употреба на Python и вклучува неколку моќни и стандардни библиотеки за справување со задачи како што се обработка на слики, развој на графички интерфејс и логика за одземање на позадина. Главно користените библиотеки се:

- OpenCV (cv2): (Open Source Computer Vision Library) е софтверска библиотека со отворен код за компјутерска визија и машинско учење. Обезбедува огромен број алатки за обработка на слики, компјутерска визија и операции во реално време. Ја избравме поради нејзиниот богат сет на функции за обработка на слики и видео, како и високи перформанси за манипулација со слики во реално време. Користен за:
  - Читање и зачувување слики: cv2.imread(), cv2.imwrite()
  - Интеграција со веб-камера: cv2.VideoCapture()
  - Конверзија на простор на бои: cv2.cvtColor() за BGR во RGB и ракување со транспарентност
  - Алфа-блендирање и транспарентност: cv2.split(), cv2.merge(), cv2.multiply(), cv2.add()
  - Компатибилност со GUI: Се користи како бекенд за обработка и подготовка на слики пред да се прикажат во Tkinter GUI.

- Tkinter: е стандардна библиотека GUI (Графички кориснички интерфејс) за Python. Овозможува креирање десктоп апликации со интерактивни компоненти како копчиња, паѓачки менија итн. го избравме бидејќи е едноставен за употреба и вграден со Python (не е потребна дополнителна инсталација), како и беспрекорната поддршка за прикажување слики со користење на PIL.ImageTk. Користен за:
    - Распоред на интерфејсот: Креиран е главниот прозорец, рамката за прикажување слики и сите контролни елементи (копчиња, паѓачки менија).
    - Поставување и преземање слики: Користени се filedialog.askopenfilename() и filedialog.asksaveasfilename() за избор на датотеки.
    - Повратни информации од корисниците: Користени се messagebox.showinfo() и messagebox.showerror() за ажурирања на статусот.

- Pillow (PIL.Image и ImageTk): е популарна библиотека за слики во Python која нуди можности за уредување слики во Python преку PIL модулот. Го избравме бидејќи се интегрира непречено со Tkinter, обезбедува едноставни алатки за промена на големината и конвертирање на слики, но најбитно бидејќи е неопходен за прикажување на слики обработени од OpenCV во Tkinter (бидејќи OpenCV користи BGR формат, а Tkinter бара формат компатибилен со PhotoImage). Користен за:
    - Конвертирање на слики од OpenCV во слики компатибилни со Tkinter користејќи Image.fromarray() и ImageTk.PhotoImage.
    - Промена на големината на сликите за GUI приказ користејќи го методот resize().

- NumPy (numpy): е основниот пакет за научно пресметување во Python. Обезбедува поддршка за големи повеќедимензионални низи и матрици, заедно со широка колекција на математички операции. Го избравме бидејќи овозможува ракување со податоци од слики како низи и е потребно за повеќето OpenCV операции. Користен за:
    - Аритметика на слики и блендирање: Се користи за директно манипулирање со вредностите на пикселите.
    - Создавање празни бели позадини: np.ones_like() за мешање слики.
    - Операции на алфа канал: Се користи за нормализирање, скалирање и манипулирање со транспарентноста.

- MediaPipe (mediapipe): е рамка од Google за градење мултимодални „pipelines“ за машинско учење. Вклучува претходно обучени модели за задачи како што се следење на раце, откривање на објекти, мрежа на лица и сегментација. Го избравме бидејќи обезбедува моќен и лесен модел на сегментација (Selfie Сегментација) кој може да работи во реално време, како и за одземање на позадина во реално време со помош на веб-камера. Користен за:
    - Сегментација во реално време преку mp.solutions.selfie_segmentation.
    - Пристап до маската за сегментација за разликување на преден објект од позадина.
    - Комбинација на маската со операции на OpenCV за динамичко отстранување или замена на позадината за време на користењето на веб-камерата.

- Threading: е стандарден Python модул за паралелно извршување на повеќе нишки (задачи). Го избравме бидејќи е неопходен за снимање и обработка на видео во реално време, така што графичкиот кориснички интерфејс (GUI) не се замрзнува за време на пристап до веб-камерата. Користен како:
    - threading.Thread() за да го стартување на преносот на веб-камерата во реално време во позадина додека главната јамка на апликацијата продолжува да работи.

- RemBg е библиотека во Python и алатка од командна линија што користи модели за длабоко учење за автоматско отстранување на позадината од сликите. Изградена е врз основа на ONNX Runtime и најчесто ја користи архитектурата U^2-Net - конволуциона невронска мрежа специјализирана за откривање на истакнати објекти. Го избравме rembg за оваа апликација затоа што е еден од најдобрите методи за Автоматско одземање, еден од нашите методи, исто така дава висококвалитетни резултати – благодарение на U^2-Net, може да детектира фини рабови. Има доста едноставна интеграција – потребна е само една функција за користење:
    - remove(input_bytes) Прифаќа бајти на слики (на пр., PNG, JPEG) и враќа бајти со отстранета позадина.

Сите овие библиотеки ни го олеснуваат процесот за креирање на нашите методи за Background subtraction, кои ќе бидат дополнително објаснати. Сепак, изборот на соодветните библиотеки за оваа демо апликација беше долг процес со оглед на одредени критериуми:

- Ефикасност на ресурсите: една од клучните предности на оваа апликација е нејзината ниска потрошувачка на ресурси, што ја прави достапна за корисниците без оглед на моќноста на нивниот уред.
- Прецизна и разновидна обработка: апликацијата овозможува прецизен избор на објекти, овозможувајќи им на корисниците сами да ги дефинираат објектите од интерес.
- Обработка на видео во реално време: избраниот модел е дизајниран за брза обработка на податоци, овозможувајќи одземање на позадината да се извршува во реално време.

Метод 1: Автоматско одземање

Овој метод нуди автоматско одземање на позадината користејќи ја горенаведената библиотека rembg, моќна алатка што користи модели на машинско учење за отстранување на позадината од сликите. Особено е корисна кога сакаме да го изолираме главниот објект (обично лице или производ) од позадината на една слика, без потреба од зелен екран или рачна анотација. За процесирање на слики RemBg ги користи библиотеките PIL(Python imaging library), pymatting, cv2, numpy, onnxruntime и scipy.ndimage. PIL примарно се употребува за процесирањето на сликите додека pymatting се употребува за алфа маторање, односно процесирање на транспарентност за добивање екстракции со висок квалитет.

За обработка на сликите, RemBg користи повеќе помошни библиотеки како што се PIL (Python Imaging Library), pymatting, OpenCV (cv2), NumPy, onnxruntime и scipy.ndimage.

- PIL се користи како главен алат за ракување со слики (вчитување, менување димензии, конвертирање итн.),
- Pymatting е задолжена за алфа матирање – техника која управува со транспарентноста со цел да се добијат реалистични и остри екстракции на објектите.

Комбинацијата од овие библиотеки овозможува висококвалитетно и прецизно процесирање на слики, со што RemBg станува оптимален избор за автоматизирано отстранување на позадина во различни апликации. Всушност се состои од една главна фунцкија, која повикува неколку помошни функции за да се постигне прецизна изолација на објектот од сликата:

```
output_image = remove(img_data)
```

Ова е основната операција што ја отстранува позадината користејќи го претходно обучениот модел во rembg. Функцијата remove() го детектира објектот во преден план (на пр., лице) и создава слика каде што позадината е или отстранета или е направена транспарентна. Резултатот е поток од бајти на слика (во PNG формат) со транспарентност во позадината.

Внатрешно, процесот функционира на следниов начин:

- Декодирање на слика: Го конвертира бајт потокот во тензорски/сличен формат читлив од моделот, односно ако влезот е во форма на бајти (наместо веќе отворена слика), тој се конвертира во слика со помош на PIL.Image за да може да се процесира понатаму.

- Извршување на модел на сегментација (U^2-Net): Моделот пресметува мапа на истакнатост на сликата. Го идентификува објектот што најмногу „привлекува внимание“ (преден план), односно

- Генерирање на маска: Се создава бинарна маска која визуелно ги одделува објектот и позадината, поставувајќи темели за понатамошна сегментација. Предниот план е целосно непросирен, а позадината станува транспарентна

- Примена на маска: Позадината или се отстранува (просирна) или се заменува, во зависност од конфигурацијата (ние користиме транспарентност).

- Вратен резултат: Резултатот се кодира назад во PNG формат и се враќа како бајти.

Останати клучни чекори и фунцкии за методот- Автоматско одземање:

1. Бидејќи функцијата remove() од rembg бара влезната слика да биде во бинарен (бајт) формат, со ова се отвора сликата на дадената патека во режим на read-binary и ја чита целата нејзина содржина во меморијата.

```
with open(img_path, "rb") as img_file:
    img_data = img_file.read()
```

2. Бидејќи remove() враќа сурови бајти на сликата, ние користиме io.BytesIO() за да ја прочитаме во меморијата и да ја завиткаме како објект сличен на датотека.

   Image.open() ја отвора сликата од меморијата.

   .convert("RGBA") осигурува дека сликата има алфа канал (транспарентност), што е важно за отстранување на позадината.

```
img_pil = Image.open(io.BytesIO(output_image)).convert("RGBA")
```

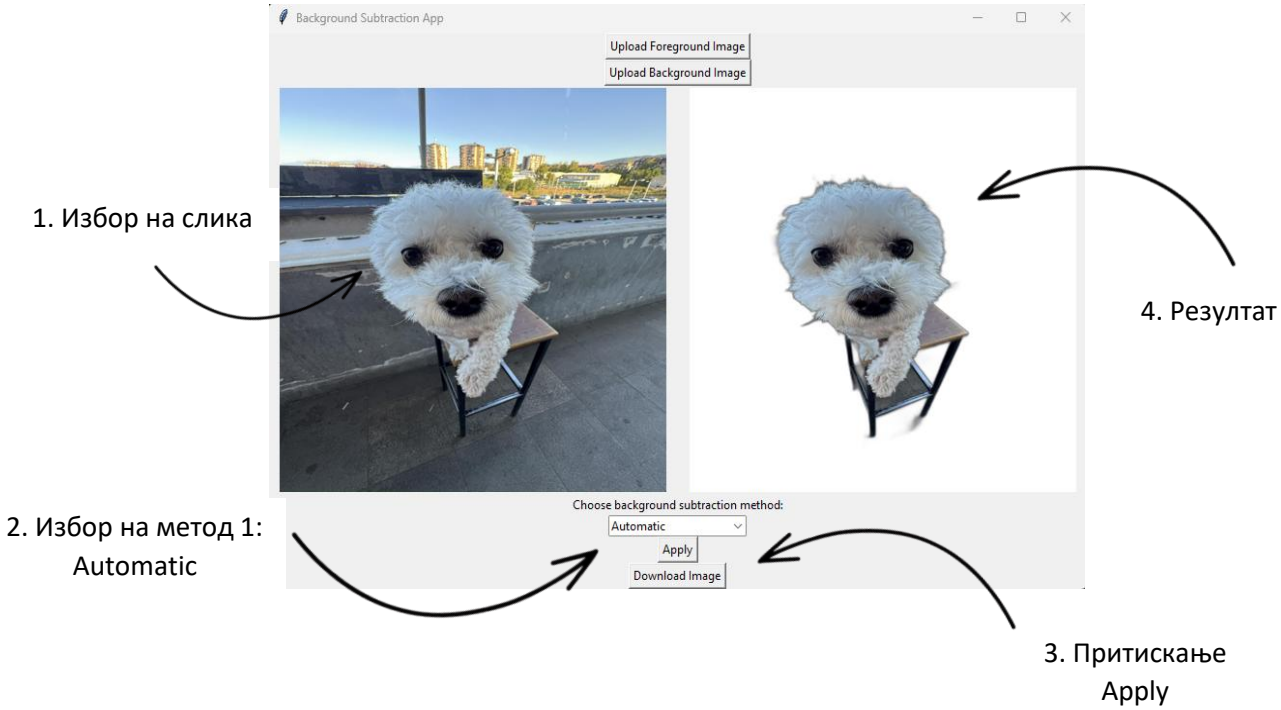
3. OpenCV користи BGR(A) формат за слики, додека PIL користи RGB(A). За да го користиме излезот со OpenCV, го конвертираме.

   np.array(img_pil): Ја конвертира сликата во NumPy низа.

   cv2.cvtColor(..., cv2.COLOR_RGBA2BGRA): Го конвертира просторот на бои од RGBA на PIL во BGRA на OpenCV.

```
result = cv2.cvtColor(np.array(img_pil), cv2.COLOR_RGBA2BGRA)
```

Пример:



Метод 2: Одземање со избор на квадрат

Методот за избор на квадрат му овозможува на корисникот рачно да нацрта правоаголник околу објектот што сака да го задржи на сликата (т.е. преден план). Врз основа на оваа избрана област, се отстранува позадината користејќи го алгоритмот GrabCut на OpenCV.

Тек на процесот:

1. Вчитување и подготовка на сликата:
   - Сликата се чита преку cv2.imread(), по што се креира копија која се користи за визуелизација на тековните промени при интеракција со корисникот, со цел да не се вршат промени директно врз оригиналната слика.

```
img = cv2.imread(img_new)
display_img = img.copy()
```

2. Интеракција со глувче - Функција за повратен повик на глувчето:

```
def select_subject(event, x, y, flags, param):
```

   - Оваа функција ја обработува интеракцијата на корисникот со глувчето при:

А) Клик на глувчето:

```
if event == cv2.EVENT_LBUTTONDOWN:
    drawing = True
    ix, iy = x, y
    mask = np.zeros(img.shape[:2], np.uint8)
```

- Започнува цртање на правоаголникот.

- Се иницијализираа маската (сите пиксели првично се означени како позадина).

Б) Движење на глувчето:

```
elif event == cv2.EVENT_MOUSEMOVE:
    if drawing:
        display_img = img.copy()
        cv2.rectangle(display_img, (ix, iy), (x, y), (0, 255, 0), 2)
        cv2.imshow('Image', display_img)
```

- Додека корисникот го влече глувчето, се црта зелен правоаголник во реално време.

В) Ослободување на глувчето:

```
elif event == cv2.EVENT_LBUTTONUP:
    drawing = False
    rect = (min(ix, x), min(iy, y), max(ix, x), max(iy, y))
    cv2.rectangle(display_img, (ix, iy), (x, y), (0, 255, 0), 2)
    cv2.imshow('Image', display_img)

    x1, y1, x2, y2 = rect
    mask[y1:y2, x1:x2] = cv2.GC_PR_FGD
    mask[mask == 0] = cv2.GC_BGD

    cv2.grabCut(img, mask, rect, bgdModel, fgdModel, 10, cv2.GC_INIT_WITH_RECT)
```

- Пуштено глувче значи финализирање на правоаголникот.

- Пикселите во избраниот правоаголник се дефинирани како преден план (cv2.GC_PR_FGD).

- Другите пиксели се позадина (cv2.GC_BGD).

- На крај се применува grabCut() користејќи го овој правоаголник како иницијализација.

GrabCut() е алгоритам за сегментација на слика што ги одделува пикселите во преден план и позадина врз основа на почетен правоаголник и итеративно ја подобрува маската.

3. Пост-обработка – Креирање на алфа-каналот

```
mask2 = np.where((mask == cv2.GC_FGD) | (mask == cv2.GC_PR_FGD), 1, 0).astype('uint8')
```

Се конвертира излезната маска од GrabCut во бинарна маска: се задржува дефинираниот преден план и се игнорира позадината.

```
b, g, r = cv2.split(img)
alpha = mask2 * 255
alpha = cv2.GaussianBlur(alpha, (5, 5), 0)
result = cv2.merge((b, g, r, alpha))
```

-Се делат каналите на сликата.
-Маската се множи со 255 за да стане вистински алфа канали се заматува со цел да се измазнат рабовите.
- Додава алфа канал за да ја направи позадината транспарентна

- На крај се спојуваат каналите во RGBA формат.

Пример:

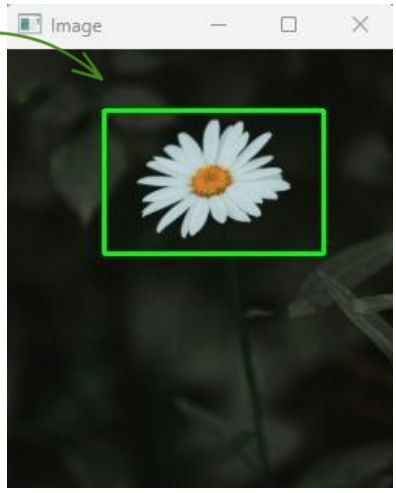
1. Избор на слика

2. Избор на метод 2:

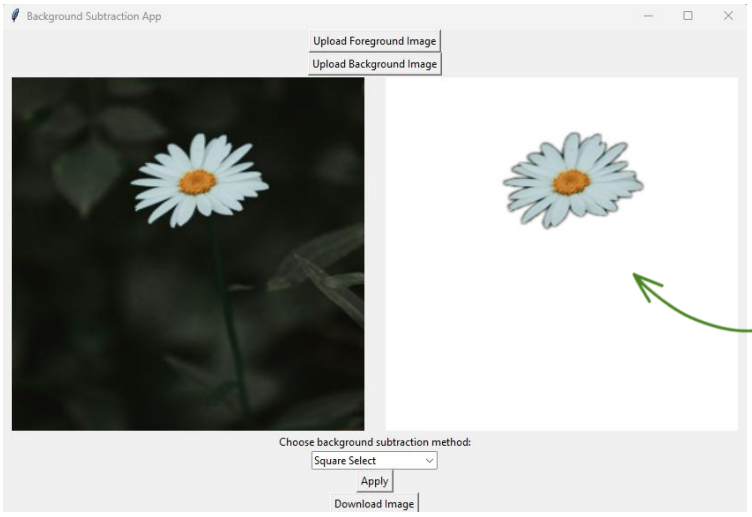
Селекција со квадрат

3. Притискање Apply

4. Исцртување на правоаголникот околу посакуваниот дел



5. Внес на буквата Q означува крај на исцртувањето



6. Резултат

Метод 3: Одземање со избор на четка

Методот со избор на четка му овозможува на корисникот рачно да го „наслика“ објектот што сака да го задржи на сликата со помош на глувчето, како четка. Овој насликан регион потоа се третира како преден план, а позадината се отстранува со помош на алгоритмот GrabCut на OpenCV со маска дефинирана од корисникот наместо правоаголник. Всушност Овој метод претставува унапредена верзија на селекција и е наменет за комплексни позадини каде класичната правоаголна селекција не е доволна.

Тек на процесот, неговите клучни чекори:

1. Функција за селекција со четка:

```
def select_subject(event, x, y, flags, param):  1 usage
    global drawing, mask, display_img
```

Ова функција се повикува при движење на глувчето, на речиси идентичен начин како метод 2:

- Означувањето започнува кога корисникот ќе притисни лев клик

```
if event == cv2.EVENT_LBUTTONDOWN:
    drawing = True
```

- При движење со притиснат клик (cv2.EVENT_MOUSEMOVE):
    - Се црта круг со тековната brush_size на маската со вредност cv2.GC_PR_FGD

```
elif event == cv2.EVENT_MOUSEMOVE and drawing:
    cv2.circle(mask, (x, y), brush_size, cv2.GC_PR_FGD, -1)
```

    - Сликата (display_img) се ажурира визуелно со зелена боја на обележаните пиксели, односно се означува областа како преден план за повратна информација до корисникот

```
display_img = img.copy()
display_img[mask == cv2.GC_PR_FGD] = [0, 255, 0]
cv2.imshow('Image', display_img)
```

- Кога корисникот го пушта левиот клик (cv2.EVENT_LBUTTONUP), drawing = False

2. Функција за ажурирање на четката

```
def update_brush_size(val):  1 usage
    global brush_size
    brush_size = max(1, val)
```

Овозможува интерактивна промена на големината на четката преку cv2.createTrackbar.

```
cv2.createTrackbar( trackbarName: 'Brush Size', windowName: 'Image', brush_size, count: 50, update_brush_size)
```

3. Извршување на логиката GrabCut во посебна нишка :

```
def background_subtraction_method_3(img_new):  1 usage
    thread = threading.Thread(target=_run_grabcut, args=(img_new,))
    thread.start()
    thread.join()
    return result
```

Функцијата користи нитка (thread) за да ја одвои обработката и дозволи UI ефикасно да работи.

result е финалната RGBA слика со отстранета позадина.

Со помош на grabCut и маска базирана на интеракција со четка, корисникот добива полу-автоматски систем со човечка флексибилност и алгоритамска моќ.

4. Основниот метод – _run_grabcut

4.1) Се вчитува сликата и иницијализира маскара како целосно непозната.

```
img = cv2.imread(img_new)
display_img = img.copy()
mask = np.zeros(img.shape[:2], np.uint8)
```

4.2) Јамка на интеракција со корисникот- корисникот означува со четката додека не притисне „q“ за да го финализира изборот

```
while True:
    cv2.imshow('Image', display_img)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
```

4.3) Примена на GrabCut:

- Се подготвува маската со означување на недопрените области како позадина.

```
mask[mask == 0] = cv2.GC_BGD
mask[mask == cv2.GC_PR_FGD] = cv2.GC_PR_FGD

try:
    cv2.grabCut(img, mask, None, bgdModel, fgdModel, 10, cv2.GC_INIT_WITH_MASK)
```

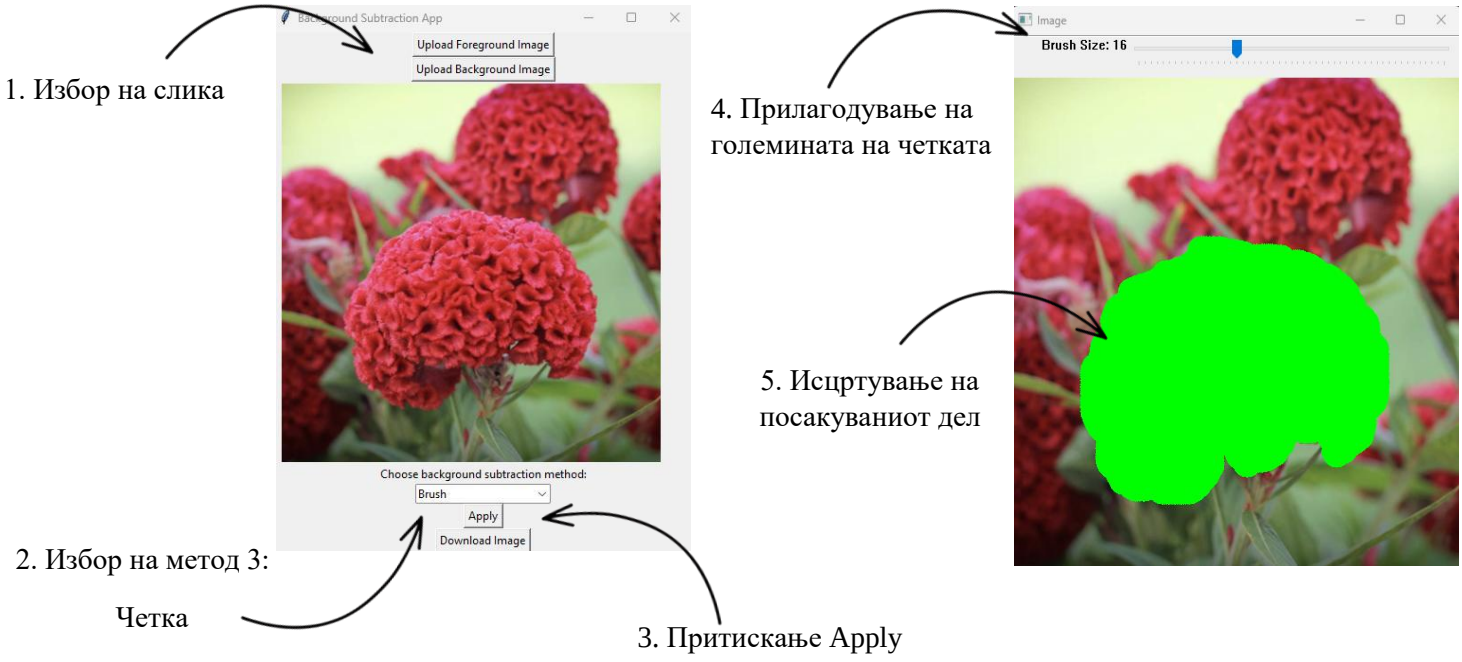
- Се користи обоената маска наместо правоаголник за сегментирање на сликата.

5. Додавање алфа канал- Се креира конечната RGBA слика каде што позадината е транспарентна.

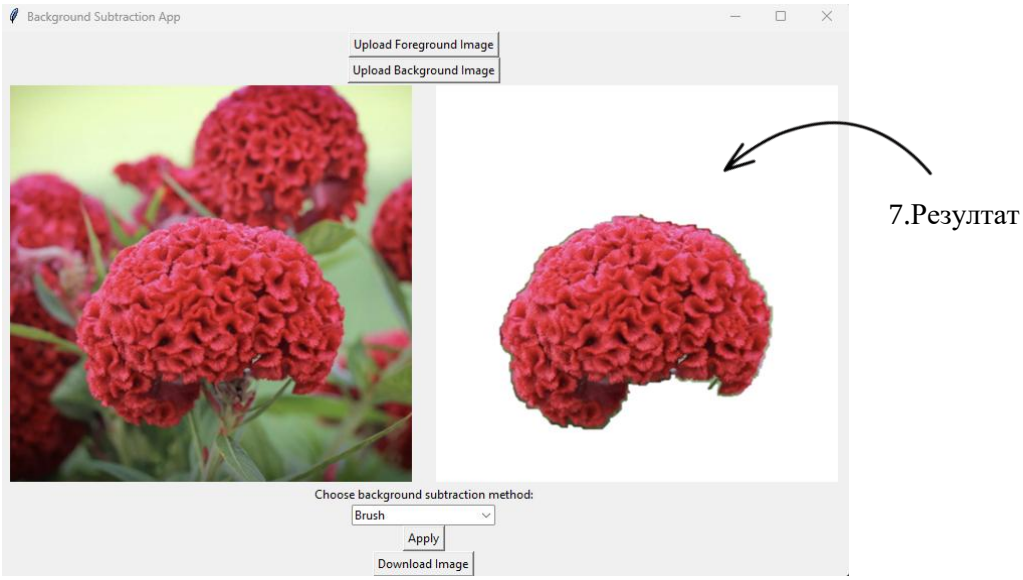
```
b, g, r = cv2.split(img)
alpha = cv2.GaussianBlur(mask2 * 255, (3, 3), 0)
result = cv2.merge((b, g, r, alpha))
```

Покрај овие клучни чекори додадени се одредени „финеси“ со цел подобар резултат, одноесно отстранување на позадината. Вклучуваат: cv2.dilate / cv2.erode за измазнување на маската во преден план, cv2.Canny за изострување на рабовите, cv2.connectedComponentsWithStats за отстранување мали бучни делови.

Пример:



6. Внес на буквата Q означува крај на исцртувањето



Метод 4: Одземање на видео позадина во реално време

Овој метод извршува отстранување и замена на позадина во реално време користејќи ја веб-камерата како влезен поток. Се потпира на моделот Selfie Segmentation на MediaPipe за да го идентификува човечкиот субјект во кадарот и да ја замени позадината со прилагодена слика (обезбедена како патека на аргументот).

Selfie Segmentation моделот (од MediaPipe) претставува лесен и ефикасен модел за сегментација на луѓе во слика или видео. Секој пиксел во влезната рамка добива вредност која означува дали припаѓа на објектот во преден план или на позадината. Резултатот е сегментациска маска која се користи за понатамошна обработка.

Маската се претвора во бинарна слика, при што пикселите над одреден праг (обично 0.1 до 0.5) се сметаат за објект. Се добива двобојна маска: една за објектот (корисникот), друга за позадината.

Новата позадина (слика прикачена од корисникот) се прилагодува на димензиите на видео-рамката со cv2.resize(). Комбинацијата се изведува со np.where() според бинарната маска:

- Пикселите што припаѓаат на објектот се зачувуваат,
- Останатите се заменуваат со пиксели од новата позадина.

Резултатот е реално време видео каде позадината е заменета, а објектот останува непроменет.

Клучните чекори во имплементација:

1. Иницијализација и вчитување на слика во позадина

```
mp_selfie_segmentation = mp.solutions.selfie_segmentation
selfie_segmentation = mp_selfie_segmentation.SelfieSegmentation(model_selection=1)
```

- Користење на SelfieSegmentation.

- Враќа маска за сегментација - низа каде што вредностите на пикселите претставуваат колку е веројатно пикселот да му припаѓа на лицето (наспроти позадината).

```
background_image_path = path
background_image = cv2.imread(background_image_path)
```

- Ја вчитува сликата за замена на позадината користејќи OpenCV.

2. Снимање од веб-камера

```
cap = cv2.VideoCapture(0)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
```

- Ја отвора стандардната веб-камера (0 е обично вградената камера), при што ќе се добиваат видео рамки во реално време.
- Континуирано снима рамки од веб-камерата сè додека корисникот не престане.

3. OpenCV дава рамки во BGR, но MediaPipe очекува RGB — оттука и конверзија.

```
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

results = selfie_segmentation.process(rgb_frame)
mask = results.segmentation_mask

condition = mask > 0.5
```

segmentation_mask: Ова е 2D низа со вредности помеѓу 0 и 1:

Вредности поблиску до 1: лице (преден план)

Вредности поблиску до 0: позадина

4. Клучната линија за замена на позадина:

```
output_image = np.where(condition[:, :, None], frame, resized_background)
```

Ако условот е True (т.е. лице), се користи оригиналниот пиксел на рамката.

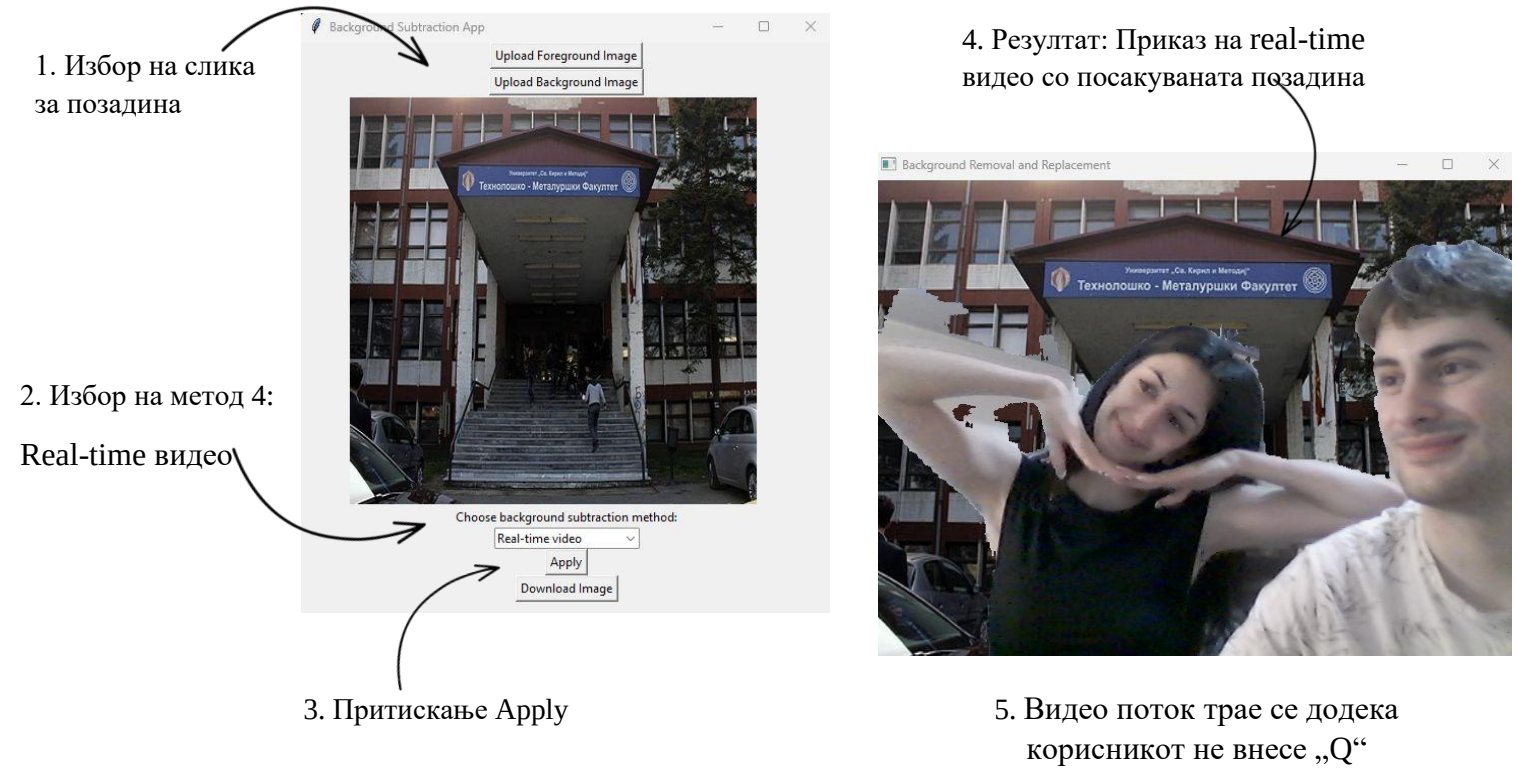
Во спротивно, се користи resized_background, односно избраната позадина скалирана за да одговара на големината на рамката на веб-камерата.

5. Приказ на изменетиот видео поток во реално време:

```
cv2.imshow('Background Removal and Replacement', output_image)
```

Се додека не се внесе “Q”

Пример:



Метод 5: Додавање на позадина на одредена слика

Овој метод е всушност комбинација на сите претходни методи, односно му овозможува на корисникот да постави слика од преден план со транспарентна позадина (алфа канал), постави нова слика од позадина. Првата слика (од преден план) може да се добие со било кои од првите 3 методи. Откако корисникот ќе ги внеси двете посакувани слики, овој метод автоматски ги спојува двете — заменувајќи ја позадината со алфа транспарентноста на предниот план. Ова е различно од претходните методи бидејќи претпоставува дека позадината е веќе отстранета и само ја заменува со алфа спојување.

Методот „Замени позадина“ обезбедува практичен и визуелно влијателен начин, не само за отстранување на несаканите позадини, туку и беспрекорно да интегрирање нови, прилагодени позадини во сликите. За разлика од другите методи кои првенствено се фокусираат на отстранување на позадината, овој метод нагласува замена, давајќи им на корисниците поголема креативна контрола и флексибилност.

Имплементација на клучните чекори:

1. Повик на методот

```
def replace_background(foreground_img, background_img):  1 usage
    if foreground_img.shape[2] != 4:
        raise ValueError("Foreground image must have an alpha channel")
```

Оваа функција очекува:

foreground_img: слика со 4 канали (вклучувајќи алфа)

backgound_img: стандардна слика со 3 канали

- Доколку предниот дел нема алфа канал (маска за транспарентност), тогаш се прикажува грешка.

2. Екстракт на алфа-каналот и нормализирање на истиот

```
b, g, r, a = cv2.split(foreground_img)
alpha = a.astype(float) / 255.0
alpha = cv2.merge([alpha] * 3)
```

Ги дели 4-те канали: B, G, R, A (алфа).

Конвертира алфа од 0–255 во 0–1 (потребно за спојување).

Ја дуплира алфа-вредноста во 3 канали за да се совпадне со RGB форматот.

3. Спојување (блендирање) на сликите:

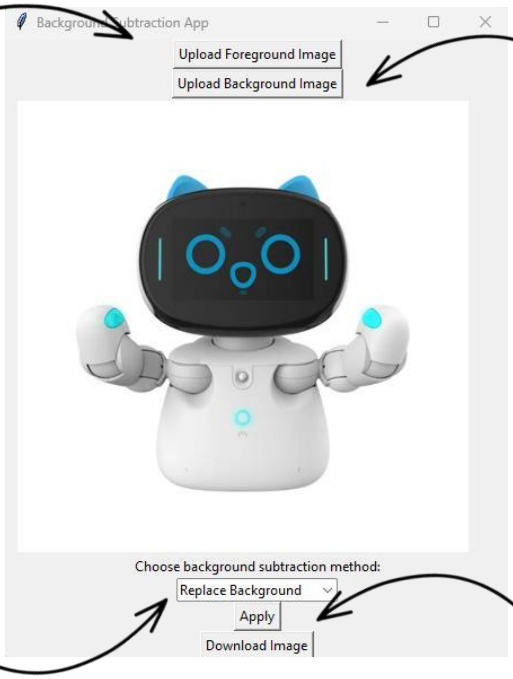
```
blended = cv2.multiply(alpha, foreground_rgb) + cv2.multiply(1 - alpha, background_rgb)
blended = blended.astype(np.uint8)
```

Формулата за спојување на алфа:

blended_pixel = alpha * foreground_pixel + (1 - alpha) * background_pixel

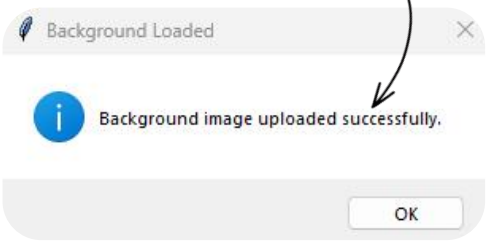
Ова непречено ги спојува двете слики, задржувајќи ги непросирните делови во предниот дел и заменувајќи ги просирните со позадината.

Пример:



1. Избор на слика со транспарентна позадина како Foreground слика

2. Избор на слика за позадина како Background слика

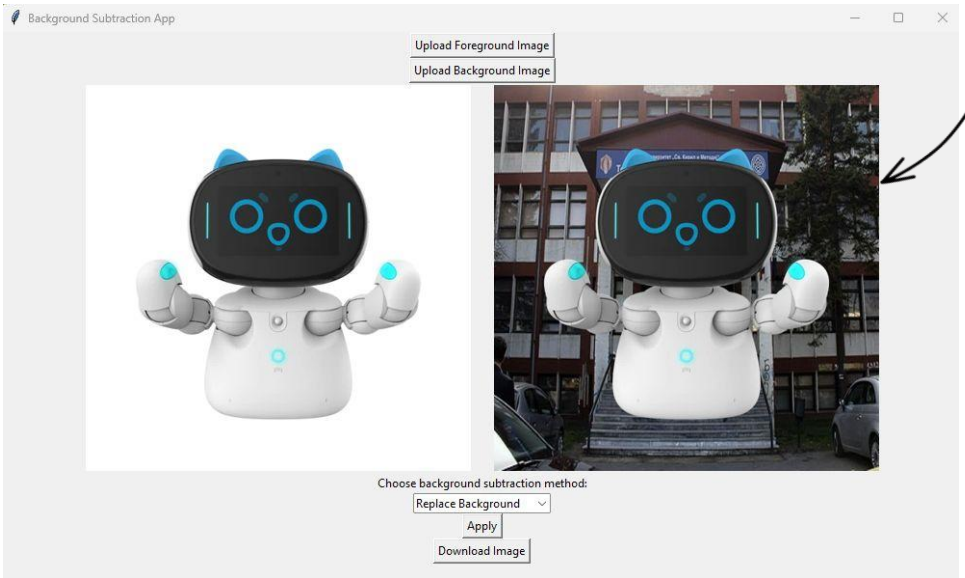


3. Избор на метод 3:

Замена на позадина

4. Притискање Apply

5. Резултат



Користена литература:

https://www.geeksforgeeks.org/background-subtraction-opencv/

https://docs.python.org/3/library/tkinter.html

https://ai.google.dev/edge/mediapipe/solutions/vision/face_detector/python

https://learnopencv.com/background-subtraction-with-opencv-and-bgs-libraries/

https://pillow.readthedocs.io/en/stable/reference/ImageTk.html