Sturdy-Octo-Disco-Adding-Sunglasses-for-a-Cool-New-Look

Sturdy Octo Disco is a fun project that adds sunglasses to photos using image processing.

Welcome to Sturdy Octo Disco, a fun and creative project designed to overlay sunglasses on individual passport photos! This repository demonstrates how to use image processing techniques to create a playful transformation, making ordinary photos look extraordinary. Whether you're a beginner exploring computer vision or just looking for a quirky project to try, this is for you!

Features:

- Detects the face in an image.
- Places a stylish sunglass overlay perfectly on the face.
- Works seamlessly with individual passport-size photos.
- Customizable for different sunglasses styles or photo types.

Technologies Used:

- Python
- OpenCV for image processing
- Numpy for array manipulations

How to Use:

1. Clone this repository.
2. Add your passport-sized photo to the `images` folder.
3. Run the script to see your "cool" transformation!

Applications:

- Learning basic image processing techniques.
- Adding flair to your photos for fun.
- Practicing computer vision workflows.

Feel free to fork, contribute, or customize this project for your creative needs!

Program

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
# Load the face image
img = cv2.imread("C:/ANU DOCUMENTS/passport size_phot.jpg")
plt.imshow(img[:,:,::-1]); plt.title("Face")

# Load the sunglass image
glassPNG = cv2.imread("C:/Users/admin/Pictures/Screenshots/Screenshot 2025-04-30 135519.png"

# Resize the sunglass image to fit on the face
glassPNG = cv2.resize(glassPNG, (180, 80))
print("Image Dimension =", glassPNG.shape)

# Check if alpha channel exists, if not add one
if glassPNG.shape[-1] == 3:
    glassPNG = cv2.cvtColor(glassPNG, cv2.COLOR_BGR2BGRA)

# Split into color and alpha channels
glassBGR = glassPNG[:, :, :3]
glassMask1 = glassPNG[:, :, 3]

# Resize glassMask to match eyeROI size
eyeROI = img[165:245, 125:305]  # Adjusted ROI
glassMask1 = cv2.resize(glassMask1, (eyeROI.shape[1], eyeROI.shape[0]))
glassBGR = cv2.resize(glassBGR, (eyeROI.shape[1], eyeROI.shape[0]))

# Display sunglass color channels and alpha channel
plt.figure(figsize=[15, 15])
plt.subplot(121); plt.imshow(glassBGR[:,:,::-1]); plt.title('Sunglass Color channels')
plt.subplot(122); plt.imshow(glassMask1, cmap='gray'); plt.title('Sunglass Alpha channel')

# Create mask for blending
glassMask = cv2.merge((glassMask1, glassMask1, glassMask1))
glassMask = glassMask.astype(np.float32) / 255  # Convert to float for blending

# Masked eye region
faceWithGlassesArithmetic = img.copy()
eyeROI = faceWithGlassesArithmetic[165:245, 125:305]

# Ensure data types match
eyeROI = eyeROI.astype(np.float32) / 255
glassBGR = glassBGR.astype(np.float32) / 255

# Apply blending
maskedEye = cv2.multiply(eyeROI, (1 - glassMask))
maskedGlass = cv2.multiply(glassBGR, glassMask)
eyeRoiFinal = cv2.add(maskedEye, maskedGlass)

# Convert back to uint8
eyeRoiFinal = (eyeRoiFinal * 255).astype(np.uint8)

# Display masked and blended regions
plt.figure(figsize=[20, 20])
plt.subplot(131); plt.imshow(maskedEye[...,::-1]); plt.title("Masked Eye Region")
```

```
plt.subplot(132); plt.imshow(maskedGlass[...,::-1]); plt.title("Masked Sunglass Region")
plt.subplot(133); plt.imshow(eyeRoiFinal[...,::-1]); plt.title("Augmented Eye and Sunglass")

# Final result with blending
faceWithGlassesArithmetic[165:245, 125:305] = eyeRoiFinal
plt.figure(figsize=[10, 10])
plt.subplot(121); plt.imshow(img[:,:,::-1]); plt.title("Original Image")
plt.subplot(122); plt.imshow(faceWithGlassesArithmetic[:,:,::-1]); plt.title("With Sunglasse
```

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Load the face image
img = cv2.imread("C:/ANU DOCUMENTS/passport size_phot.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Load the sunglasses image with alpha channel
glassPNG = cv2.imread("C:/Users/admin/Pictures/Screenshots/Screenshot 2025-04-30 135519.png"

# Check alpha channel and convert if needed
if glassPNG.shape[-1] == 3:
    glassPNG = cv2.cvtColor(glassPNG, cv2.COLOR_BGR2BGRA)

# Load Haar cascade for eye detection
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

# Detect eyes
eyes = eye_cascade.detectMultiScale(gray, 1.3, 5)

# If two eyes are detected, proceed
if len(eyes) >= 2:
    # Sort eyes by x-axis to get left and right
    eyes = sorted(eyes[:2], key=lambda x: x[0])
    (x1, y1, w1, h1), (x2, y2, w2, h2) = eyes

    # Compute sunglass position and size
    x = x1
    y = min(y1, y2)
    w = (x2 + w2) - x1
    h = max(h1, h2)

    # Resize sunglasses to cover both eyes
    glassResized = cv2.resize(glassPNG, (w + 40, h + 40))  # Adjusted width/height

    # Split channels
    glassRGB = glassResized[:, :, :3]
    glassMask = glassResized[:, :, 3] / 255.0
```

```
    glassInvMask = 1.0 - glassMask

    # Get region of interest (ROI) on face
    y1 = max(0, y - 20)
    x1 = max(0, x - 20)
    y2 = y1 + glassResized.shape[0]
    x2 = x1 + glassResized.shape[1]

    roi = img[y1:y2, x1:x2].astype(float)

    # Resize ROI if needed
    if roi.shape[0] != glassResized.shape[0] or roi.shape[1] != glassResized.shape[1]:
        roi = cv2.resize(roi, (glassResized.shape[1], glassResized.shape[0]))

    # Blend sunglass with ROI
    for c in range(3):
        roi[:, :, c] = (glassMask * glassRGB[:, :, c] + glassInvMask * roi[:, :, c])

    # Replace blended region back to face
    img[y1:y2, x1:x2] = roi.astype(np.uint8)

# Show final result
plt.figure(figsize=(8, 8))
plt.imshow(img[:, :, ::-1])
plt.title("With Sunglasses")
plt.axis('off')
plt.show()
```

```
Out[3]: Text(0.5, 1.0, 'With Sunglasses')
```