Project Name :- “The Mysterious Cloak”

Team Name :- “The Pixel Manipulators”

Abstract:

"The Mysterious Cloak" is a project that leverages computer vision and image processing techniques to create an illusion of invisibility similar to the mythical invisibility cloak seen in various works of fiction. The project captures video from a webcam, identifies a specific colour range (in this case, red), and replaces it with the background scene, giving the impression that objects of that colour are invisible.

Overview:

The project utilizes the Flask web framework to create a web application interface. Users can access the application through a web browser, which displays the live video feed from the webcam with the invisibility effect applied in real-time. The core functionality involves capturing the background scene, detecting and masking the specified colour range, and overlaying the background onto the video feed in place of the detected colour.

Advantage:

1. **Novelty:** The project offers a fun and intriguing way to experiment with computer vision techniques.

2. **Educational Purpose:** It provides a hands-on opportunity for learning about image processing, colour detection, and masking.

3. **Interactive Experience:** Users can interact with the project in real-time through the web interface, making it engaging and entertaining.

Disadvantage:

1. **Limited Colour Range:** The current implementation only supports masking a specific colour range (red). This limitation restricts the versatility of the application

How to overcome with this Disadvantage:

1. **Multi-colour Support:** Extend the application to support masking multiple colours or even arbitrary objects, enhancing its versatility and practicality.

Example of Real-World Application and Explanation:

The concept of an invisibility cloak, though fictional, has inspired various real-world applications across different fields:

1. **Surveillance and Security:** In the realm of surveillance and security, technologies inspired by invisibility cloaks could be used to conceal cameras or sensors, allowing for covert monitoring without arousing suspicion. This could be particularly useful in sensitive or high-security environments where discreet observation is necessary.

2. **Entertainment and Special Effects:** In the entertainment industry, the concept of invisibility cloaks has been utilized to create visually stunning special effects in movies, television shows, and theatrical productions. By incorporating similar technologies, filmmakers and animators can bring fantastical scenes to life and captivate audiences with immersive storytelling.

3. **Medical Imaging:** In the field of medical imaging, advanced camouflage techniques inspired by invisibility cloaks could be employed to enhance the visibility of certain anatomical structures or to improve the contrast between tissues in diagnostic imaging modalities such as MRI or CT scans, leading to more accurate diagnoses and better patient outcomes.

Overall, while the notion of invisibility cloaks may have originated in the realm of fiction, ongoing advancements in technology continue to blur the lines between fantasy and reality, opening up new possibilities for innovation and discovery across a wide range of industries and applications.

HOW OUR CODE IT WORKS EXPLAIN THIS FIRST TO LAST :-

```
from flask import Flask, render_template, Response

import cv2

import numpy as np

import time
```

- Imports necessary modules: `Flask` for creating a web application, `render_template` for rendering HTML templates, `Response` for generating HTTP responses, `cv2` for OpenCV library which is used for computer vision tasks, `numpy` for numerical operations, and `time` for time-related functions.

```
app = Flask(__name__)
```

- Creates a Flask application instance.

```
def invisibility_cloak():
    cap = cv2.VideoCapture(0)
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter('invisibility_cloak.avi', fourcc, 20.0, (640, 480))
    time.sleep(2)
    background = 0  # capturing background
    for i in range(30):
        ret, background = cap.read()  # capturing image
```

- Defines a function `invisibility_cloak()` which captures the background from the webcam. It initializes a video capture object `cap`, sets up a video writer `out`, waits for 2 seconds to ensure proper initialization, and then captures 30 frames from the webcam to obtain the background.

```
    while (cap.isOpened()):
        ret, img = cap.read()
```

- Starts an infinite loop to continuously capture frames from the webcam.

```
    if not ret:
        break
```

- Breaks out of the loop if there's no frame captured.

```
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask1 = cv2.inRange(hsv, lower_red, upper_red)

    lower_red = np.array([170, 120, 70])
    upper_red = np.array([180, 255, 255])
    mask2 = cv2.inRange(hsv, lower_red, upper_red)

    mask1 = mask1 + mask2  # OR
```

- Converts the BGR image (`img`) to HSV color space, then creates masks `mask1` and `mask2` to detect red color in two ranges.

```
    mask1 = cv2.morphologyEx(mask1, cv2.MORPH_OPEN, np.ones((3, 3), np.uint8), iterations=2)
    mask2 = cv2.morphologyEx(mask1, cv2.MORPH_DILATE, np.ones((3, 3), np.uint8), iterations=1)
    mask2 = cv2.bitwise_not(mask1)
```

- Performs morphological operations to remove noise from the masks and refine them. Also, computes the inverse of `mask1` and stores it in `mask2`.

```
    res1 = cv2.bitwise_and(background, background, mask=mask1)
    res2 = cv2.bitwise_and(img, img, mask=mask2)
    final_output = cv2.addWeighted(res1, 1, res2, 1, 0)
```

- Uses bitwise operations to isolate the parts of the image corresponding to the red color and the background, then combines them to create the final output.

```
        ret, buffer = cv2.imencode('.jpg', final_output)
        final_output = buffer.tobytes()

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + final_output + b'\r\n')
```

- Encodes the final output image as a JPEG and yields it as part of a multipart response for video streaming.

```
@app.route('/')
def index():
    return render_template('index.html')
```

- Defines a route for the home page and renders an HTML template named `index.html`.

```
@app.route('/video_feed')
def video_feed():
    return Response(invisibility_cloak(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

- Defines a route for streaming video and returns a Response object that streams the output of the `invisibility_cloak()` generator function with the specified MIME type.

```
if __name__ == '__main__':
    app.run(debug=True)
```

- Runs the Flask application in debug mode if the script is executed directly.