Computer Vision Project 2 Report

I. Introduction

Objectives and Goals

The primary objective of this project is to develop a computer vision system capable of tracking key elements within the "Codenames" game, such as player movements, card placements and detect which card has been played. The system will be able to identify the game board, player's hands, and cards, and track their movements throughout the game.

The system will also be able to identify when a player has made a guess. The system will be able to identify when a team has lost the game.

IV. Data Set Description

We've recorded 9 videos depicting the game in different levels of difficulty for the problem. The difficulty is based on light variations and/or camera movement. The clips are named accordingly and you can view the input data as well as prepared outputs in the following link.

Image Preprocessing Steps

For red and blue cards we decided to work with raw images, but for assassin and neutral cards we had to do some preprocessing, since those colors are hard to detect in HSV range as well as other common ranges. The preprocessing steps for images of those cards are described below.

1. **Loading and Resizing:**
   - The reference images (assassin, neutral1, and neutral2) are loaded rotated and downscaled to a resolution of (176, 120) pixels using.

```
assassin = cv2.imread('data/assassin.jpg', cv2.IMREAD_COLOR)
assassin = cv2.rotate(assassin, cv2.ROTATE_90_COUNTERCLOCKWISE)
assassin = cv2.resize(assassin, (176, 120))
```

2. **Background Removal using HSV Color Space:**
   - The images are converted to the HSV color space.

- Masks are created for each image by thresholding the HSV values to extract the desired color range. We prepared the photos so that it's easier to extract the color range.

```
assassin_hsv = cv2.cvtColor(assassin, cv2.COLOR_BGR2HSV)
assassin_mask = cv2.inRange(assassin_hsv, (0, 0, 0), (100, 100, 100))
```

3. **Morphological Operations (Opening):**
   - Morphological opening is applied to refine the masks and remove unwanted noise, using a kernel of size (7, 7).

```
assassin_mask = proper_opening(assassin_mask, np.ones((7, 7), np.uint8))
```

4. **Mask Application and Cropping:**
   - New images are created by keeping only the pixels corresponding to the identified regions using the refined masks.
   - Background is cropped from the images to focus only on the relevant content.

```
new_assassin[assassin_mask == 255] = assassin[assassin_mask == 255]
```

5. **Grayscale Conversion:**
   As grayscale is better for keypoint detection, we converted the images to grayscale.
   - The preprocessed images are converted to grayscale using `cv2.cvtColor` to simplify further image analysis tasks.

```
assassin_gray = cv2.cvtColor(assassin, cv2.COLOR_BGR2GRAY)
```

The resulting preprocessed images, both in color and grayscale, are then ready for further analysis.

Methodology

You can see our methodology in the `do_everything()` function in `codenames.ipynb` . (Function name is WIP).

Authors

Samuel Janas 151927
Bruno Urbaniak 151955