UNO CARDS RECOGNISER

Coursework 2 "Intelligent Sensing for Robotics PDE4434"

The goal of this coursework is to create a program that can recognize UNO cards. The program will recognize cards in an image or frame using data from a file (data set) or a regular camera (webcam).

In this project I am submitting the following files:

I. Uno-colour detection – Jupyter notebook.

II. Uno-number_shape detection – Jupyter notebook.

III. Uno-combination (colour+shape) using the dataset only – Jupyter notebook.

IV. Uno-combination (colour+shape) using the webcam only – Jupyter notebook and Python script.

V. Uno-combination (colour+shape) using the dataset & webcam – (webcam-NEW.py) - python script.

##

Notes:

- **"CARDS" is the name of the folder that contains the data set "UNO cards" and it should be saved in the same directory as the python script.**
- **Make sure to have/download the below libraries: [1],[2]**

Library name	Installing using pip (python) [1]	Installing using conda (Anaconda) [2]
glob	pip install glob2	conda install -c conda-forge glob2
cv2	pip install opencv-python	conda install -c conda-forge opencv
matplotlib	pip install matplotlib	conda install -c conda-forge matplotlib
numpy	pip install numpy	conda install -c anaconda numpy
random	pip install random2	Already exist
argparse	pip install argparse	conda install -c conda-forge argparse

[1] https://pypi.org/

[2] https://anaconda.org/

I. Uno-Colour Detection

In this stage, I followed the instructions below:

1. Loading the data set images into Jupyter.
   Load the images found in CARDS/*.

2. For each dataset image, crop out the card
   To locate the cards in the image, edge detection is used. This is accomplished using an adaptive threshold. After that, any external contour is chosen as a possible card contour. The card contour is the inner contour. An orientated bounding box is used to approximate the contour (OBB). The four corner points then were utilized to rotate and trim the card using a perspective transform.

3. Split the dataset into training and testing (80% / 20%)
   Split the dataset into training and testing. Ideally, we would have 70%/30% but this could be too much where a single colour is only in testing, thus a ratio of 80% / 20% is picked to be safe.

4. For the training dataset, does the colour detection
   - Resize the image to a very small image using NEAREST interpolation
     The resizing is mainly done for speed up. K-means which is used afterwards is slow and we should avoid processing too big images. It is important to choose NEAREST interpolation so that colours are not mixed when downsizing. Resize it to have a width of 10.

   - Do a k-means (K=2) to find white colour and uno card colour
     First the image is reshaped into a single column vector, where each component is the colour of a pixel. Then apply k-means to find 2 centres. The images should only have the colour white, and the UNO colour.

   - Pick colour which is farthest from white, should pick uno colour
     The colour detection has to be robust against light changes. We choose here to pick the HUE which is better than picking RGB directly.

   - Transform colour to HSV and pick hue as dominant colour

5. For the test dataset, test against the training dataset

   Does the same step as the training data to find the dominant colour. Then find the training data which has the closest dominant colour and pick its label.

   - Find dominant colour as explained above
   - Compare dominant colour with train dataset dominant colour
   - Pick closest hue value in train dataset
   - Make sure that train and test label for colour match

II. Uno-Number/Shape Detection

In this stage, I followed the instructions below:

1. Load the dataset images
2. For each dataset image, crop out the card

To locate the cards in the image, edge detection is used. This is accomplished using an adaptive threshold.

After that, any external contour is chosen as a possible card contour. The inner contour is picked as the card contour.

An orientated bounding box is used to approximate the contour (OBB). The four corner points are then utilized to realign and trim the card picture using a perspective transform.

3. Normalize image size
   Resizing/Scaling: the ration/scale = 1.5, width = 400 & height = 600

4. Split the dataset into training and testing
   For every number, pick 3 for training and 1 for testing randomly.

5. For the training dataset, does the number detection
   - Do an adaptive threshold to find edges in card
   - Find all the external contour in the card
   - Pick the contour which is the closest to the center
   - Save number contour
6. For the test dataset, test against the training dataset
   - Find number contour as explained above
   - Do a contour matching against the contour in the train dataset
   - Make sure that train and test label for number match

III. Uno-combination (colour+shape) using the dataset only

In this stage you have to upload an image from the UNO CARDS. (code line 4 – Jupyter notebook)

ex: ("CARDS/"*" ") Replace the "*" with the image name + '.jpeg'

Example names:

* ZERO_B

* TWO_B_DRAW

* SKIP_Y

 # NOTE: you can add new images to the data set folder "should be similar to the training data set with a black or dark background" and then upload them for testing, like the image below:

"I have tested up to 2 cards in one image"

Here I combined the colour and shape detection and used the following functions:

- find_cards
- find_dominant_colour
- find_number_contour
- Find card for each image
- Normalize image sizes
- Train for colour and number/shape
- Test the data

IV. Uno-combination (colour+shape) using the webcam only

I followed the previous procedure but only changed the way of receiving the data, as I am feeding the system here by using the webcam.

V. Uno-combination (colour+shape) using the dataset & webcam – python script

After combining (colour+number/shape) I have added the following arguments:

- ✓ **('--images', help="Input CARDS")**
- ✓ **('--camdevice', type=int, default=0, help="Camera number")**
- ✓ **('--mode', choices=['livecam', 'staticimg'], default='livecam', help="Mode")**
- ✓ **('--input', type=str, default='', help="Input image for detection")**

++

The following methods are showing how to run the python script:

A. Example of how to run the python script and using static images (dataset):

python webcam-static.py --mode staticimg --images CARDS --input CARDS/ONE_Y.jpeg

B. Example of how to run the python script and using the live cam (webcam):

python webcam-static.py --mode livecam --camdevice 1 --images CARDS