REAL-TIME PLAYING CARDS RECOGNITION SYSTEM

Work by:
Dona Shenika Divyanjalee Ranasinghe, Ispravnic Malina, Flavia Alessandra Rodríguez Pereda

INTRODUCTION

This project aims to develop a practical and efficient system capable of recognizing playing cards in real time. The primary objective is to create a robust solution that can be seamlessly integrated as a main piece in more complex systems to aid in various real-life scenarios, particularly in gaming and surveillance environments.

By integrating this computer vision system, automated systems can be developed to analyze the cards dealt with, calculate probabilities, and provide real-time assistance to players or dealers. This could enhance decision-making processes, improve game flow, and offer insights into optimal strategies, thus elevating the overall gaming experience.

Furthermore, this project holds considerable promise in the domain of card counting and cheating detection within casinos and other gambling environments. By continuously monitoring the cards being dealt and played, the proposed system can identify irregularities or patterns indicative of foul play. This can serve as an invaluable tool for surveillance systems, providing an additional layer of security and ensuring compliance with fair gaming practices.

EXISTING APPROACHES

When developing a real-time playing card detector system, various approaches can be considered, each offering distinct advantages and trade-offs. Below, we discuss several commonly used methods in computer vision for similar tasks, highlighting their potential benefits.

1. **Convolutional Neural Networks (CNNs)**: They have demonstrated superior performance in many image recognition tasks due to their ability to learn complex features directly from data. CNNs are highly accurate and robust to variations in scale, rotation, and lighting conditions, making them an ideal choice for playing card recognition. CNNs automatically learn relevant features during the training process, eliminating the need for handcrafted features. Popular CNN architectures such as ResNet, VGG, and MobileNet can be employed, and transfer learning techniques can be utilized to adapt pre-trained models (e.g., on ImageNet) to the playing card recognition task. This approach can significantly reduce both the training time and the amount of labeled data required.

2. **YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector):** YOLO and SSD are object detection models designed for real-time applications. They offer end-to-end detection, providing bounding boxes and class probabilities in a single forward pass through the network. These models are known for their excellent balance between detection speed and accuracy. YOLO and SSD are particularly advantageous for tasks requiring rapid and accurate localization and recognition of multiple objects within an image. Training these models on a dataset of annotated playing cards can enable real-time detection and recognition, which might be more challenging with traditional methods.

3. **Deep Learning-Based Feature Matching:** It mixes the power of CNNs to extract highly discriminative features, which can then be used with various matching techniques. Deep features are more robust and can improve recognition accuracy. This method involves extracting deep features using a pre-trained CNN and building a descriptor database. Nearest neighbor search algorithms, such as FLANN, can then be used to match features between the input frame and the reference database. This approach combines the robustness of deep learning with the flexibility of feature matching.

4. **Hybrid Approaches**: They combine traditional methods with deep learning to leverage the strengths of both. For instance, initial key point detection and description can be performed using ORB or SIFT, followed by a deep learning model to refine the recognition results. Another hybrid strategy could involve using a CNN for feature extraction and simpler classification methods, thus combining the robustness of CNN features with the simplicity of traditional classification techniques. These methods offer adaptability and can be tailored to the specific requirements and constraints of the application.

OUR APPROACH

The base algorithm used to create this project is the Bag of Visual Words (BoVW) model, which is a popular approach for object recognition and image classification tasks in computer vision.

The BoVW model involves the following main steps:

1. Feature Detection and Description: In this code, the ORB algorithm is used for detecting key points and computing descriptors for both the sample images (reference database) and the video frames from the webcam
2. Codebook Generation: We did not implement the generation of a codebook (visual vocabulary), which is a crucial step in the BoVW model. Instead, we used the descriptors from the sample images as the reference database for matching
3. Feature Encoding: In the BoVW model, the descriptors from an image are typically encoded using the codebook to create a histogram or bag-of-visual-words representation. However, in this project, the feature encoding step is skipped, and the descriptors are directly matched against the reference database
4. Matching and Classification: The code uses the Brute-Force Matcher and k-Nearest Neighbour(kNN) matching to find the best matches between the descriptors from the video frame and the descriptors in the reference database. The class with the maximum number of good matches (filtered using the Nearest Neighbour Distance Ratio technique) is selected as the predicted class for the frame

While this project doesn't strictly follow the complete BoVW pipeline, it borrows the core concept of representing images using local feature descriptors and matching them against a reference database for object recognition and classification.

The BoVW model is a popular choice for object recognition tasks because it provides a stable and efficient way to represent and match image features, even in the presence of scale and rotation changes or visual impediments. However, the project in this case simplifies the BoVW model by skipping the codebook generation and feature encoding steps, which can be computationally expensive and may not be necessary for the specific task of playing card recognition. In the end, this resulted in a more streamlined and efficient implementation of our objective.

DEVELOPMENT

1. CREATION OF A DESCRIPTOR DATABASE

In this project, the creation of descriptors is a critical step in developing an efficient computer vision system for real-time playing card recognition. We utilize the OpenCV and NumPy libraries to manage and process the images. The initial phase involves loading sample images of playing cards, specifically an Ace and a Six, and storing them in a list to serve as the reference database for matching against video frames.

To facilitate the feature extraction process, we define a list of classes representing the different ranks of playing cards.

We then implement a function named *"describe"* which is tasked with creating descriptors by taking a list of images as input using the ORB (Oriented FAST and Rotated BRIEF) feature detector and descriptor. ORB combines the FAST key point detector with the BRIEF descriptor to provide an efficient and robust solution for feature extraction.

ORB makes use of the FAST (Features from Accelerated Segment Test) algorithm to detect key points in the images. FAST is designed to quickly identify corners or interest points in the image, making it highly suitable for real-time applications. ORB then assigns an orientation to each key point based on the intensity centroid of the local image patch. This step

ensures that the descriptors are invariant to rotation, which is crucial when recognizing playing cards that may appear at various angles.

After detecting and orienting the key points, ORB uses the BRIEF (Binary Robust Independent Elementary Features) algorithm to compute the descriptors. BRIEF generates binary strings by comparing the intensities of pairs of points in a smoothed image patch around each key point. The result is a compact and distinctive binary vector that encapsulates the local image information. The combination of FAST and BRIEF in ORB results in a descriptor that is both fast to compute and robust to changes in scale and rotation. This efficiency makes ORB particularly well-suited for real-time playing card recognition, where speed and accuracy are one of our main focuses.

The generated descriptors are then stored in a list, creating a reference database that can be used for matching against video frames. While alternative methods such as SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) were considered, ORB was chosen due to its superior balance of speed and accuracy. SIFT and SURF, although robust and accurate, are computationally more intensive and less suited for real-time applications.

By utilizing these image processing techniques, we ensure that our reference database is robust and capable of accurately matching playing cards in various real-life scenarios. This foundational step is crucial for the subsequent development and testing phases of the project, as it underpins the system's ability to perform real-time card recognition with high accuracy.

2. BUILDING THE CLASSIFICATION FUNCTION

We create a function to classify the cards detected on the frames we get in real-time named *objclsf*

To achieve object classification by matching features extracted from a video frame against the descriptors in the reference database, we utilize the ORB feature detector and descriptor. Employing it to extract key points and descriptors from the real-time input frames.

Next, a Brute-Force Matcher (BFMatcher) is implemented. The BFMatcher compares each descriptor from the input frame with every

descriptor in the reference database, calculating a distance (typically Hamming distance for binary descriptors like those from ORB) between each pair. This exhaustive search is computationally intensive but ensures the best matches are found by considering all possible pairs.

To enhance the matching process, we employ k-Nearest Neighbor (kNN) matching. In kNN matching, for each descriptor in the input frame, the k closest descriptors in the database are found. Here, the k=2 parameter specifies that the two nearest neighbors should be returned for each descriptor in the video frame. This step helps in identifying the most similar descriptors and is crucial for filtering out poor matches.

The function then applies a distance ratio test, which compares the distance of the closest match (d1) to the second closest match (d2). This ratio, known as the Nearest Neighbor Distance Ratio (NNDR), helps reduce false positives. The match is considered good if the ratio $\frac{d1}{d2}$ is below a certain threshold (0.75 in this case), indicating that the closest match is significantly better than the second closest one.

After filtering based on the NNDR, the function retains only the good matches. The predicted class of the frame is determined by selecting the class with the maximum number of good matches. If the number of good matches for a particular class exceeds a threshold (set to 10 in this case), the function returns the index of that class. If not, it returns “-1”, indicating no match found.

Alternative methods like template matching or simpler feature matching approaches might not offer the same level of accuracy and stability required for tasks such as playing card recognition. As these approaches often struggle with variations in scale, orientation, and lighting conditions, which are common in real-world scenarios. Therefore, the comprehensive approach using ORB with BFMatcher and kNN matching stands out for its ability to handle such challenges effectively, making it ideal for demanding applications like playing card recognition.

3. MAIN FUNCTION IMPLEMENTATION

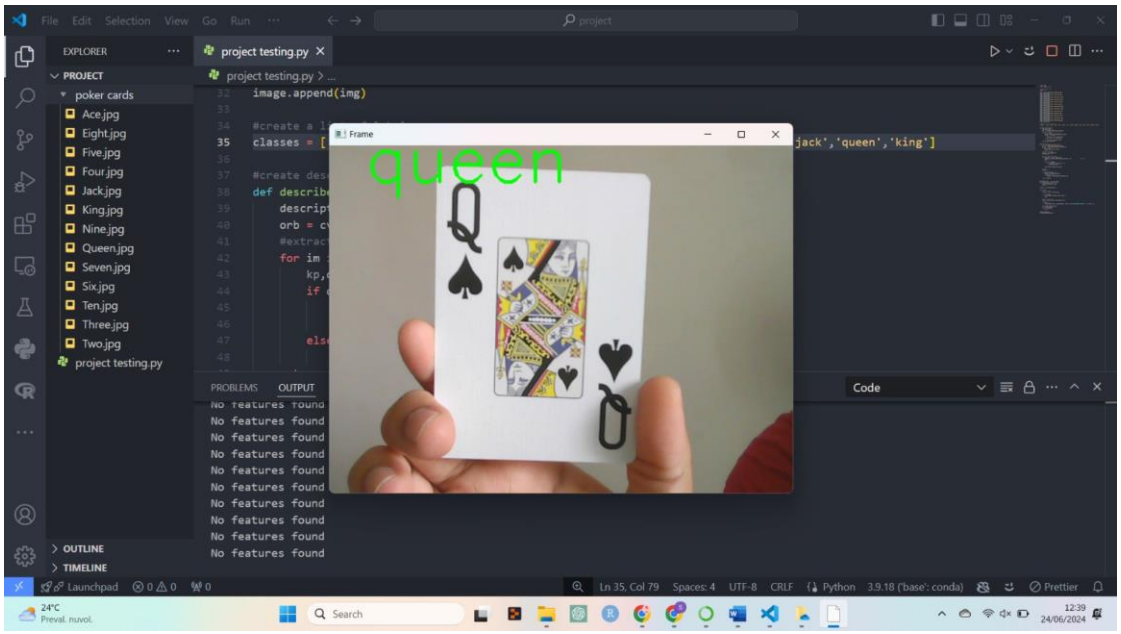
The main function of our real-time playing card recognition system begins by calling the *describe* function to create the descriptor list from the sample images loaded initially. These descriptors serve as the reference database against which incoming video frames are matched.

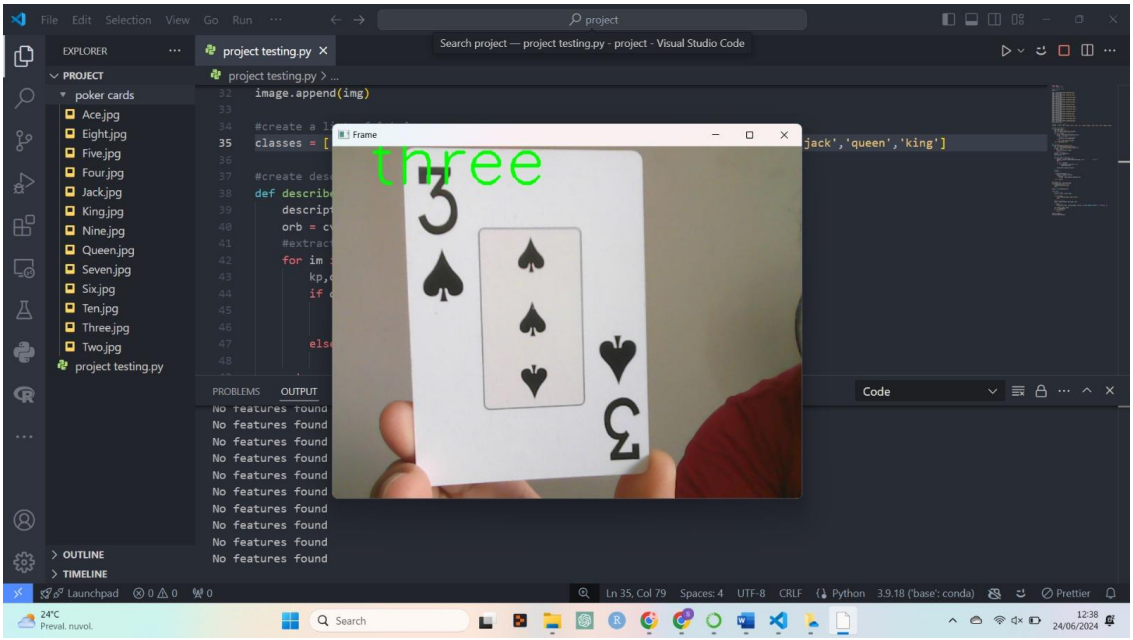
Then, we established the webcam automatically opening, thus enabling us to access the webcam feed. The code will enter a loop where it continuously reads frames from the webcam using the read method. If a frame is successfully read, it calls the *objclsf* function to perform object classification on the frame, passing the descriptor list as an argument

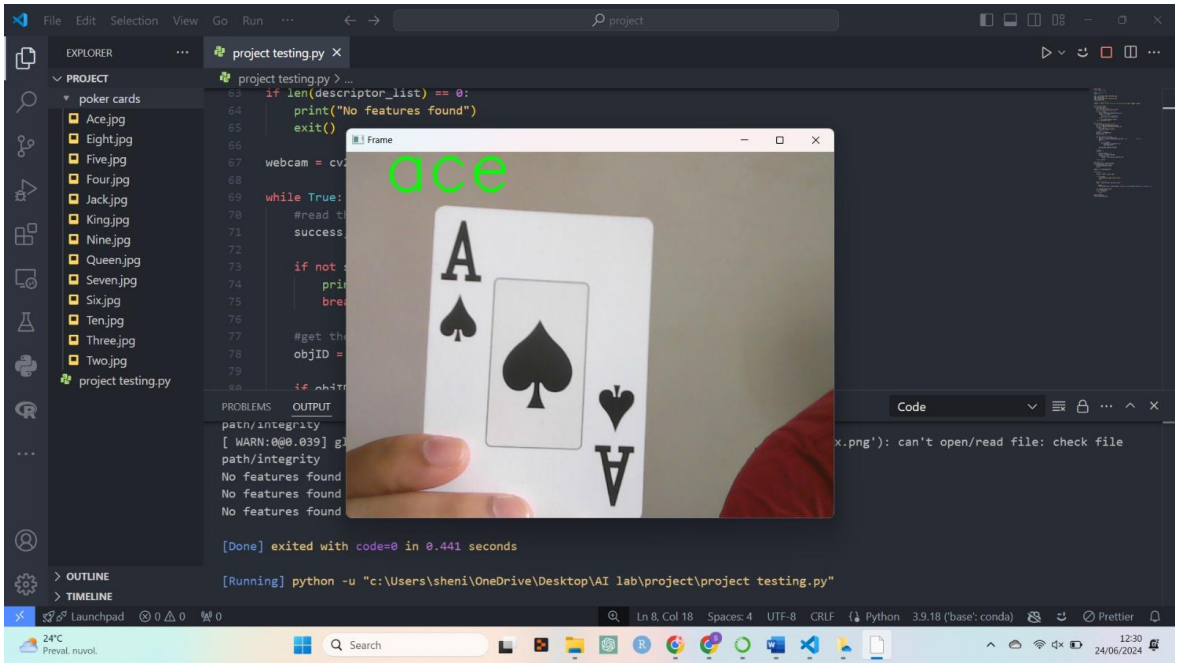
If the *objclsf* function returns a valid class ID (not -1), the code will display the correct class label (card rank) on the frame using the *putText* function from OpenCV. The frame is then instantly displayed. We enable breaking the loop by pressing a key of choice and releasing the webcam.

This approach of continuously processing video frames and displaying results in real time is efficient and practical for the task of playing card recognition. Alternative approaches that involve processing the entire video at once or using complex deep-learning models might not be as suitable for real-time applications or might require more computational resources.

SOME RESULTS







REFERENCES

E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, Spain, 2011

Mouats, Tarek & Aouf, Nabil & Nam, David & Vidas, Stephen. (2018). Performance Evaluation of Feature Detectors and Descriptors Beyond the Visible. Journal of Intelligent & Robotic Systems.

Velasco, Nancy & Mendoza, Dario & Andaluz, Víctor & Domínguez, Sergio. (2017). Anglo-American Playing Cards Identification Based on Counting Each Symbols and Scale Invariant Feature Transform (SIFT)

https://szeliski.org/Book/