COMP 6600 Final Project

Title - Object Recognition using Computer Vision

Author: Nitin Yadav

Part 1: Real-time Shape Recognition and Tracking

Abstract

This project presents a straightforward but efficient framework for monitoring common shapes in real-time video footage, including triangles, rectangles, and bubbles. The primary objective is to consistently assign and preserve IDs for every item across 153 frames, addressing issues such as identity switching, occlusion, and non-rigid movements. The project aims to utilise well-known classical computer vision algorithms, such as the Hough Circle Transform for recognizing circular objects, CIELAB color space features for better color constancy under various lighting conditions, and Canny edge detection for distinct border identification. It employs the Hungarian method for object tracking between frames, identifying objects according to color and radius similarities. This method is intended to be both interpretable and computationally efficient, in contrast to deep learning models, which frequently function as "black boxes." The initial findings, which demonstrate precise detection and consistent tracking performance, are encouraging. Visual overlays and analytics such as ID-switch counts and detection rates support these results.

Introduction

A fundamental problem in computer vision, tracking several objects across video frames has a wide range of applications in fields such as biology, surveillance, industrial monitoring, and agriculture. Traditional computer vision techniques are frequently more effective and simpler to understand than deep learning models when the objects are simple forms, such circles or rectangles. This is particularly crucial in trials with limited datasets or in real-time circumstances.

Real-time image processing, for example, can be used in:

1. Precision agriculture to help identify weeds and crops. A microcontroller can then use this data to accurately spray nutrients or herbicides, increasing productivity, preserving soil health, and cutting waste.

2. Traffic management is another useful application, where real-time video feeds may track traffic patterns and automatically modify traffic signals to ease congestion.

The methodologies employed in this project are based on a solid foundation of traditional approaches. While Hough Transform, first presented by Duda and Hart in 1972, is useful for identifying geometric shapes, Canny's edge detection algorithm (1986) is still a dependable method for determining object bounding boxes precisely. This project offers an organized, visible substitute for deep learning by integrating these tools with color-based analysis (using CIELAB space) and a deterministic tracking technique utilizing the Hungarian algorithm (Kuhn, 1955). The end product is a specialized pipeline that can accurately and consistently identify objects across frames while tracking basic moving shapes, which can translated into experimental or real-world scenarios.

Background / Related Work

This project expands upon several tried-and-true methods from traditional computer vision. One of the most important tools is Canny edge detection (Canny, 1986), which is still the preferred technique for picture preprocessing because it strikes a good compromise between noise reduction and edge sensitivity. The Hough Circle Transform, which is based on the original Hough Transform first presented by Duda and Hart in 1972 and is especially useful for recognizing circular shapes, is another crucial element. To maintain consistent object identities as they move across video frames, the project uses the Hungarian algorithm (Kuhn, 1955), a well-established solution for efficiently solving the assignment problem. For handling variations in lighting, the CIELAB color space (Sharma & Trussell, 1997) is used, as it offers perceptual uniformity and helps make color-based tracking more reliable. This project prioritizes interpretability and speed, which makes classical methods a better fit, even if contemporary tracking systems like SORT (Bewley et al., 2016) and DeepFlow (Wang et al., 2017) successfully combine traditional vision techniques with deep learning. The method employed here for efficient object tracking is further supported by teaching materials such as Stanford's CS131 course, which highlights the need of integrating form and appearance features. On the other hand, object detection frameworks such as YOLO (You Only Look Once) have been widely employed for high-speed object recognition in real-world scenarios and are made for broad purposes and end-to-end learning. Furthermore, RANSAC (Random Sample Consensus) offers a robust statistical approach for model fitting, such as detecting geometric shapes under noisy conditions. Though powerful, both YOLO and RANSAC are excluded from this implementation in favor of a more interpretable and determined pipeline.

Approach

Pipeline Overview:

1. Edge Detection

- Use Canny edge detector and convert RGB to CIELAB to isolate chromatic information for edge refinement. (Figure 2)

2. Circle Detection
    - Use Hough Circle Transform (via OpenCV) to detect circular objects in each frame.
    - Store coordinates and radius of detected circles in each frame.
3. Matching Across Frames
    - Construct a cost matrix between detections in frame t and t+1 based on:
        - Euclidean distance between centers
        - Difference in radius
    - Solve the assignment using the Hungarian algorithm for optimal object match.(figure 3)
4. Occlusion Handling
    - If a circle is not detected in the current frame, interpolated using past and next position and assign ID with the average radius.
5. Final Outputs
    - Annotated video with bounding circles in blue color, consistent IDs in black and predicted circles in red color.
    - CSV output with radii and center coordinates for all detected circles per frame.(x, y, r, ID, frame)

Matching Cost Function for circle *i* in frame *t* and circle *j* in frame *t+1*:

$$Cost_{i,j} = \alpha * (C_i - C_j)^2 + \beta * (r_i - r_j)^2$$

Where $C_i - C_j$ is the Euclidean distance between circle i and j in consecutive frames and r_i, r_j are their respective radii. $\alpha = 0.1$and $\beta = 0.9$ to ensure strict radius matching.

Results

Quantitative Evaluation Metrics:

- Tracking Accuracy: >80% of total bubbles are tracked with consistent IDs across all frames.
- ID Switches: Reduced to fewer than 30 across 1530 bubble instances.
- Detection Rate: HoughCircles detects ~87% of bubbles per frame.

Qualitative Results:

- Visual overlays show robust tracking with minimal drift or ID switching.
- CSV exports enable further analysis (e.g. coordinates of bubbles)

Part 2: Digit Recognition using CNN on MNIST Dataset

Objective

The goal of this project is to build a robust Convolutional Neural Network (CNN) that can accurately recognize handwritten digits. Using the MNIST dataset, we train and validate the model and then test it on an external image containing handwritten digits arranged in rows.

Key aims:

- Achieve high accuracy on standard MNIST test data.
- Evaluate generalization by predicting digits in a custom external image.
- Explore preprocessing steps (thresholding, padding, centering) to improve predictions on non-MNIST data.

Approach

1) Dataset: Train on the MNIST dataset (60,000 training images, 10,000 test images). Test on a custom external image containing 100 handwritten digits (10 digits × 10 rows).
2) Data Preprocessing: Normalize pixel values, reshape images, threshold external images, and segment digits.
3) Model: CNN with two Conv2D layers, MaxPooling, Flatten, Dense layer with Dropout, and Softmax output.
4) Training: 20 epochs, batch size 32, with validation each epoch.
5) Testing: Evaluate on MNIST test data and on an external image.

Implementation

Data Loading & Visualization: Loaded MNIST, normalized and reshaped images, one-hot encoded labels, and visualized samples. Below are the parameters of model building compiling and training.

Build the CNN Model

- Two Conv2D layers (32 & 64 filters, 3×3) with ReLU and MaxPooling to extract features.
- Flatten to convert features into a vector.
- Dense layer with 128 units (ReLU) + Dropout for regularization.
- Output layer with 10 units (softmax) for digit classification.

Compile the CNN Model

- optimizer='adam',

- loss='categorical_crossentropy',
- metrics=['accuracy'].

Train the CNN Model

The model is trained on the MNIST training set for 20 epochs with a batch size of 32. Validation is performed on the test set each epoch to monitor performance using accuracy metrics which is simple calculation of how many digits are correctly recognised. The model architecture is summarized in table 1 in appendix.

Results

On MNIST Test Set

- Test accuracy: ~99%
- Accuracy vs Epochs: Training and validation accuracy converge near 99%.
- Loss vs Epochs: Both losses decrease and remain low with no major gap.

On External Image

- Preprocessing segments 100 digits (10 per row).
- Total correct predictions: 97/100
- Accuracy on external image: 97%

Visual results are provided in figure 4 in appendix.

Applications

A CNN trained for handwritten digit recognition (like MNIST model) is a building block for many real-world applications like following:

- Postal Mail Sorting: Automatically recognize ZIP codes, postal codes, and address numbers to speed up mail and package sorting.
- Bank Cheque Processing: Read handwritten amounts or account numbers on cheques to reduce errors and accelerate clearing.
- Form and Document Digitization: Extract handwritten fields from scanned forms to automate data entry in various sectors.
- Automated ID/Passport Number Reading: Recognize handwritten digits on official forms to speed up verification processes.
- Financial Transactions: Read handwritten numeric inputs in banking workflows to minimize transcription errors and save time.
- Education/Exam Processing: Grade handwritten numeric answers on exams to save teachers' time and ensure consistency.

Conclusion

In conclusion, this project is aimed to apply two complementary and powerful applications of classical computer vision and deep learning techniques. In the first part, a robust real-time object tracking framework is developed to detect and consistently track simple shapes such as circles across video frames. By leveraging well-established algorithms like Canny edge detection, Hough Circle Transform, and the Hungarian assignment method, the framework achieved strong tracking accuracy with minimal ID switches and high detection rates, all while maintaining interpretability and computational efficiency. In the second part, a Convolutional Neural Network(CNN) is trained on the MNIST dataset to perform handwritten digit recognition. The model achieved near-perfect accuracy on standard MNIST test data and generalized well to external images through careful preprocessing techniques, reaching 97% accuracy.

Together, these two pipelines highlight how simple computer vision methods and deep learning techniques can be effectively combined for practical computer vision tasks, offering scalable, accurate, and explainable solutions for real-world applications ranging from industrial monitoring and precision agriculture to delivery automation and financial data processing, reducing manual effort to a great extent.

References

- http://vision.stanford.edu/teaching/cs131_fall1718/files/cs131-class-notes.pdf

- Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). *Simple online and realtime tracking.* IEEE International Conference on Image Processing (ICIP). DOI: 10.1109/ICIP.2016.7533003

- Canny, J. (1986). *A computational approach to edge detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 679–698 DOI: 10.1109/TPAMI.1986.4767851
- Deng, L. (2012). The MNIST Database of Handwritten Digit Images for Machine Learning Research. IEEE Signal Processing Magazine, 29(6), 141–142.

- Duda, R. O., & Hart, P. E. (1972). *Use of the Hough transformation to detect lines and curves in pictures.* Communications of the ACM, 15(1), 11–15. DOI: 10.1145/361237.361242
- Keras Documentation: https://keras.io

- Kuhn, H. W. (1955). *The Hungarian method for the assignment problem.* Naval Research Logistics Quarterly, 2(1-2), 83–97. DOI: 10.1002/nav.3800020109
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

- OpenAI. (2024). ChatGPT (June 2025 version) [Large language model]. https://chat.openai.com

- Sharma, G. & Trussell, H. J. (1997). *Digital color imaging.* IEEE Transactions on Image Processing, 6(7), 901–932. DOI: 10.1109/83.585269
- TensorFlow Documentation: https://www.tensorflow.org

- Wang, Y., Xu, Y., & Yuille, A. L. (2017). *DeepFlow: Large displacement optical flow with deep matching.* IEEE International Conference on Computer Vision (ICCV) DOI: 10.1109/TPAMI.2014.2343963

Appendix

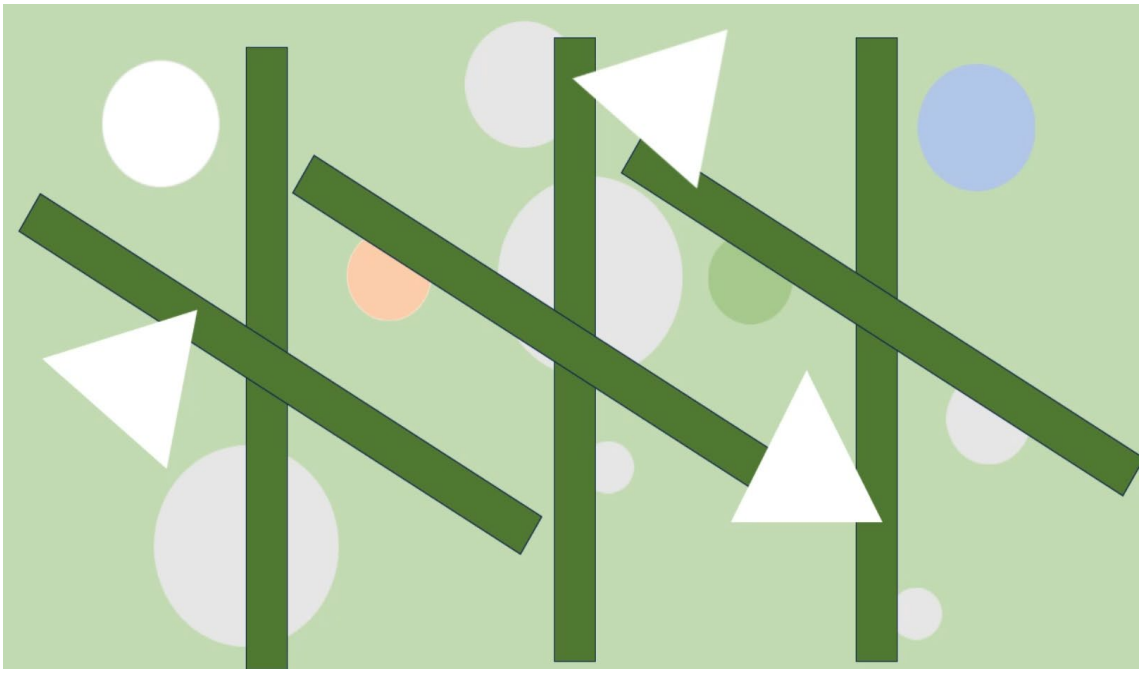

Figure 1: Sample frame of input video

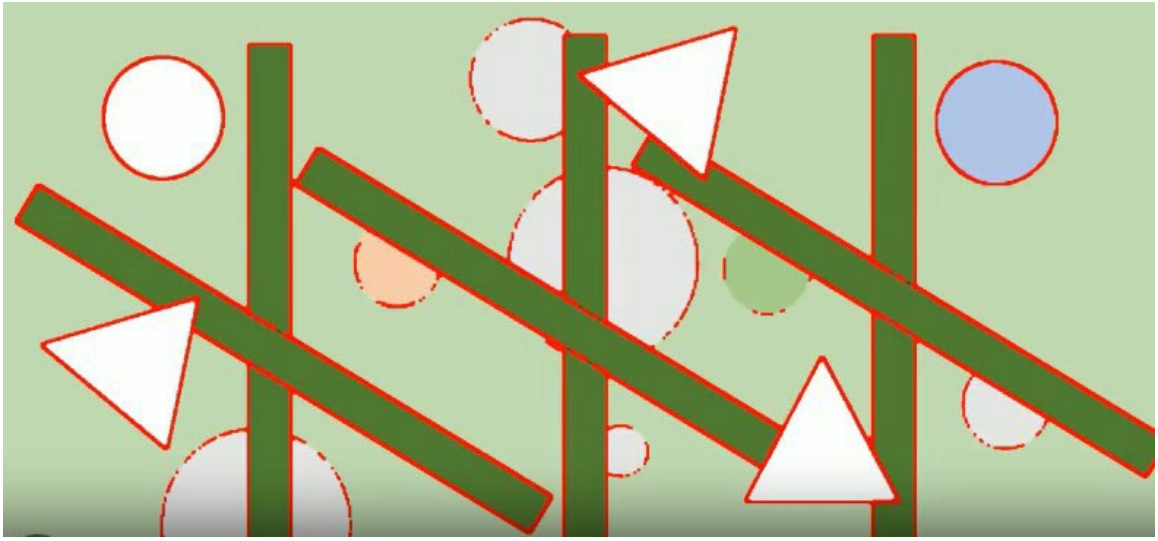

Figure 2: sample frame of edge detected video.

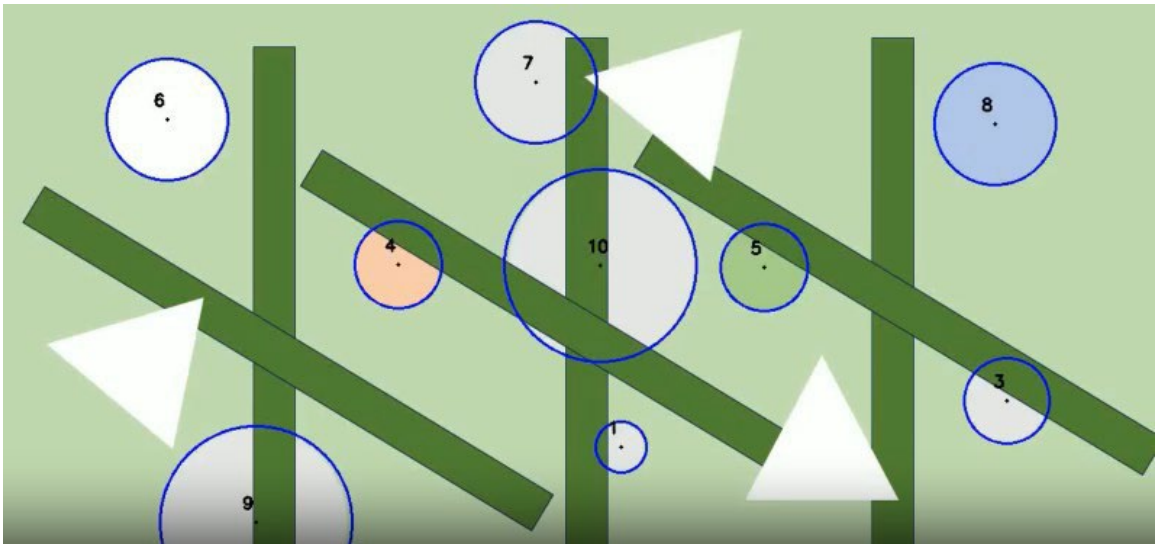

Figure 3: Sample frame of circle detected and tracking video.

Layer	Filters/Units	Kernel/Size	Activation	Notes
Conv2D	32	3x3	ReLU	Extract features
MaxPooling2D	-	2x2	-	Downsample features
Conv2D	64	3x3	ReLU	Deeper features
MaxPooling2D	-	2x2	-	Downsample features
Flatten	-	-	-	Convert to vector
Dense	128	-	ReLU	Fully connected
Dropout	0.5	-	-	Regularization
Dense	10	-	Softmax	Output class scores

Table 1: MNIST Model architecture and parameters.

Figure 4: Predictions on custom image by MNIST model.