Kecilin AI Test Code

Anugrah Aidin Yotolembah

Topic

Things that will be the main discussion

1. technology & IDE
2. Models
3. Logic Program
4. Accuracy Report
5. Output video & Conclusions

Tecnology & IDE

A. The technologies & IDE used in this program code include :

1. **OpenCV (cv2):**
   Used to record images, including reading and writing video, as well as detecting objects.

2. **Number of Py (np):**
   Used for mathematical operations and array manipulation, for example in image manipulation.

3. **TensorFlow (tf):**
   Used to download and deploy artificial neural network models to detect objects.

4. **Python**
   Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It's easy to learn, has a clear syntax, and supports a wide range of applications, from web development to scientific computing. Python's dynamic typing, extensive standard library, and cross-platform compatibility make it a popular choice among developers worldwide.

5. **Vscode**
   Visual Studio Code (VSCode) is a lightweight and versatile source code editor developed by Microsoft. It offers a streamlined interface with powerful features for coding, debugging, and version control.

6. **CSV**
   CSV (Comma-Separated Values) is a file format used to store data in a tabular form, where each row represents a record in the table, and each column is separated by a comma (,). It is a commonly used format for data exchange between various applications, especially spreadsheets like Microsoft Excel.

Models

- Model Selections : Convolutional Neural Network (CNN)

  The model contained in this program code is an artificial neural network model that has been previously drilled to detect objects. This model may have been drilled using an architecture such as a Convolutional Neural Network (CNN) or another architecture commonly used for object detection tasks.

```
# Load model deteksi objek
model_path = '/Users/didiyotolembah19gmail.com/Documents/kerja/kecilin startup/kecilin test/model/keras_model.h5'
model = tf.keras.models.load_model(model_path)
```

Logic Program

The program uses an artificial neural network model to predict the class of objects in each video frame. If the detected object is a top, the program will perform additional steps to check whether the top is in the arena, by checking whether its center coordinates are within the specified area of the arena. If the top is in the arena, the program will detect the rotation angle of the top and draw a bounding box around it.

```python
# Fungsi untuk mendeteksi kelas objek (berputar, tidak berputar, normal)
def detect_class(image, model):
    input_image = cv2.resize(image, (224, 224))
    input_image = np.expand_dims(input_image, axis=0)
    input_image = input_image / 255.0

    predictions = model.predict(input_image)
    class_index = np.argmax(predictions)
    confidence = np.max(predictions)

    # Return class index based on the model's output
    if class_index in [0, 1]:
        return class_index, confidence
    elif class_index == 2:  # Adjust the index if non-spinning top class is different
        return 2, confidence
    else:
        return None, None
```

```python
# Definisi area arena (misalnya menggunakan koordinat x, y, lebar, dan tinggi)
arena_x, arena_y, arena_width, arena_height = 100, 100, 1024, 768

# Inisialisasi penghitung gasing
spinning_tops_count = 0
non_spinning_tops_count = 0
```

Accuracy Report

In this report, the model has analyzed objects within a video frame. Each entry in the report corresponds to an object detected by the model, providing the following details:

- **Frame**: The frame number in the video where the object was identified
- **Class**: The classification of the detected object. Here, most objects are labeled as "Spinning Top," except for rows 35-38, classified as "Non-Spinning Top."
- **Accuracy**: The model's detection accuracy, expressed as a percentage.
- **Bounding Box**: The object's location in the frame, defined by the coordinates of its top-left and bottom-right corners within a bounding box.

For instance, in the initial entry of the report, the model recognized a spinning top with 84.69% accuracy in frame 1. The spinning top's bounding box is situated at coordinates (447, 568) for the top-left corner and (180, 121) for the bottom-right corner.

detection_report

Frame	Class	accuracy	Bounding Box
1.0	Spinning Top	0.8469214	(447, 568, 180, 121)
2.0	Spinning Top	0.84000075	(446, 567, 181, 122)
3.0	Spinning Top	0.93692416	(507, 578, 181, 121)
4.0	Spinning Top	0.93796873	(507, 578, 181, 121)
5.0	Spinning Top	0.9720981	(605, 550, 117, 179)
6.0	Spinning Top	0.970948	(573, 580, 180, 117)
7.0	Spinning Top	0.9811147	(672, 542, 116, 180)
8.0	Spinning Top	0.9815268	(672, 542, 116, 180)
9.0	Spinning Top	0.96875393	(691, 555, 167, 101)
10.0	Spinning Top	0.9677536	(691, 555, 167, 101)
11.0	Spinning Top	0.9434682	(671, 528, 168, 95)
12.0	Spinning Top	0.94416887	(671, 528, 168, 95)
13.0	Spinning Top	0.9617089	(656, 528, 169, 93)
14.0	Spinning Top	0.96104264	(656, 528, 169, 94)
15.0	Spinning Top	0.9160398	(683, 497, 90, 168)
16.0	Spinning Top	0.9155921	(683, 497, 90, 168)
17.0	Spinning Top	0.86455745	(667, 487, 88, 170)
18.0	Spinning Top	0.8564058	(667, 487, 88, 170)

106.0	Spinning Top	0.6771458	(580, 360, 16, 24)
106.0	Spinning Top	0.6771458	(950, 245, 12, 39)
106.0	Spinning Top	0.6771458	(933, 193, 35, 9)
107.0	Non-Spinning Top	0.5624081	(920, 229, 13, 38)
107.0	Non-Spinning Top	0.5624081	(923, 187, 26, 11)
108.0	Non-Spinning Top	0.566997	(920, 229, 13, 38)
108.0	Non-Spinning Top	0.566997	(923, 187, 26, 11)
109.0	Non-Spinning Top	0.68727684	(900, 214, 14, 43)
109.0	Non-Spinning Top	0.68727684	(902, 175, 27, 13)
110.0	Non-Spinning Top	0.6854164	(900, 214, 14, 44)
110.0	Non-Spinning Top	0.6854164	(902, 175, 27, 13)
111.0	Non-Spinning Top	0.56839526	(642, 434, 10, 14)
111.0	Non-Spinning Top	0.56839526	(885, 203, 16, 51)
112.0	Non-Spinning Top	0.568826	(642, 434, 10, 14)
112.0	Non-Spinning Top	0.568826	(885, 203, 16, 51)
113.0	Non-Spinning Top	0.5824784	(670, 444, 10, 30)
113.0	Non-Spinning Top	0.5824784	(873, 207, 13, 40)
114.0	Non-Spinning Top	0.58400065	(670, 444, 10, 30)

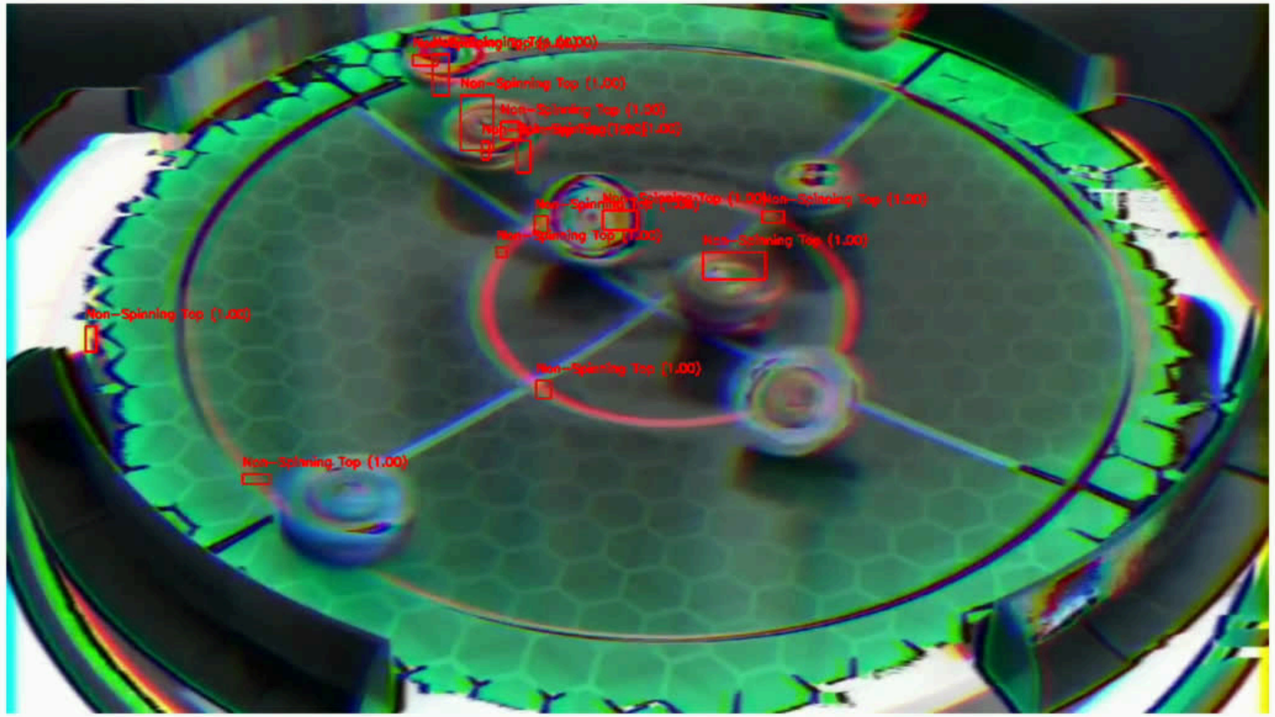
Accuracy Report

object detections from the program's results, including their accuracies and detections, will be saved into a CSV-formatted file.

```
# Buka file CSV untuk report
csv_file = open('detection_report.csv', mode='w')
csv_writer = csv.writer(csv_file)
csv_writer.writerow(['Frame', 'Class', 'accuracy', 'Bounding Box'])
```

```
# Simpan informasi ke dalam file CSV
csv_writer.writerow([cap.get(cv2.CAP_PROP_POS_FRAMES), text, confidence, (rotated_x, rotated_y, rotated_w, rotated_h)])
```

Output Video



Conclusions

1. This code aims to detect rotating and non-rotating tops from the input video.

2. Each frame of the video is captured, and detection is performed using a pre-trained neural network model.

3. The detected tops are then given a bounding box and classified as rotating or non-rotating based on the model prediction results.

4. Detection information, including frame number, object class, confidence level, and bounding box coordinates, is recorded in a CSV file.

5. Match duration is also recorded and saved in a CSV file.

6. Once the process is complete, an output video is created that displays the original frame with bounding boxes added, and information about the number of rotating and non-rotating tops is displayed on the console.

Thank You