**1. OCR Detection Robustness**

python

CopierModifier

def detect\_score(image, win\_width, win\_height, region=None):

# ...

text = pytesseract.image\_to\_string(thresh, config=config)

digits = ''.join(filter(str.isdigit, text))

try:

return int(digits)

except ValueError:

return 0

The OCR detection could be improved with:

* Consistency checks on detected values (comparison with previous values)
* Adaptive image preprocessing based on lighting conditions
* More advanced filtering to reduce noise

**2. Performance Optimization**

The program analyzes full images at each iteration, which can be slow:

python

CopierModifier

data = analyze\_game\_image(image, window\_box[2], window\_box[3], detection\_regions)

Suggestions:

* Implement a more efficient multi-threaded processing approach
* Selectively analyze regions that change frequently
* Reduce the analysis frequency (currently every second)

**3. Error Handling**

The current code contains generic try-except blocks:

python

CopierModifier

try:

# code

except Exception as e:

print(f"Error in analysis loop: {e}")

time.sleep(1)

Possible improvements:

* Handle specific exceptions based on type
* Log errors into a file instead of just printing them to the console
* Implement a more sophisticated error recovery strategy

**4. User Interface**

The current interface is functional but basic:

python

CopierModifier

tk.Label(root, text="Score:").grid(row=0, column=0, sticky="e")

tk.Entry(root, textvariable=score\_var, width=20).grid(row=0, column=1)

Potential improvements:

* A more modern interface using frameworks like PyQt or Kivy
* Graphical visualization of trends (e.g., score over time)
* More detailed information on detected elements

**5. Complex Element Detection**

Detection of trains, stations, and lines relies on simple methods:

python

CopierModifier

def detect\_stations(image, win\_width, win\_height, region=None):

# ...

peri = cv2.arcLength(cnt, True)

approx = cv2.approxPolyDP(cnt, 0.04 \* peri, True)

Improvements:

* Use machine learning algorithms (e.g., CNNs) for shape detection
* Implement object tracking between frames for consistency
* Develop a more robust approach to distinguish different station types

**6. More Flexible Configuration**

The program currently uses a simple JSON file for configuration:

python

CopierModifier

if os.path.exists(CONFIG\_FILE):

try:

with open(CONFIG\_FILE, "r") as f:

detection\_regions = json.load(f)

Possible enhancements:

* A more comprehensive configuration interface (thresholds, OCR settings, etc.)
* Configuration profiles for different screen resolutions
* Automatic calibration of detection regions

**7. Documentation & Modularity**

The code lacks detailed documentation, and some functions are quite long:

python

CopierModifier

def detect\_trains(image, win\_width, win\_height, region=None):

# Long and complex function

Suggested improvements:

* Better documentation (more detailed docstrings, examples)
* Refactoring into smaller, more modular classes and methods
* Unit tests to validate detection functions