C++ Coding Challenge: “Marker Coverage Estimator”

Task Objective:

This take-home assignment is designed to evaluate how you approach and solve real-world computer vision problems, as well as the quality and structure of your C++ code and overall solution. This assignment reflects the type of challenges you would encounter while working in a CV/AI environment and developing DeepStream-based pipelines in real-world scenarios.

Test Details:

Each input image contains exactly one Sodyo color marker. A marker is defined as a 3 × 3 grid composed of color patches. Each patch is one of the following six distinct colors: red, green, yellow, blue, magenta, or cyan.

Write a C++ console application that:

1. Receives one or more image paths as command-line arguments.
2. For each image, the app will compute the marker's bounding polygon and return the ratio (0–100%) of the marker’s area to the whole image area, rounded to the nearest integer.
3. Prints results in the format:

```
<image_file>  <coverage_percent>%
```

Example:

```
marker_01.png  83%
```

Sample Images

You will be given access to a collection of sample images to test and validate your solution.

https://sodyo.sharepoint.com/:f:/s/RD/Es6NAjVrlaZOjBtPR8GnPnIBTmg9QZ_y0dYvUawzrUtQtQ?e=tFnZfb

Technical constraints

- Language: Modern C++
- Libraries: You may use open-source, cross-platform libraries. Avoid platform-specific APIs.
- Code quality: Follow modern C++ best practices

Functional requirements

#	Requirement
FR-1	Detect exactly one 3 × 3 patch marker per image. You may assume no other 3 × 3 colored grids exist.
FR-2	Support PNG and JPEG input.
FR-3	Robust to rotation (±45 °), perspective tilt, moderate blur, and moderate color shift (see sample images).
FR-4	Runtime ≤ 200 ms for a 640 × 480 image on a laptop CPU (Intel i5 or Apple M-series).
FR-5	Exit with non-zero code if no marker is found.
FR-6	Debug mode: When enabled, the application should produce clear and informative log output to assist with development and troubleshooting.
FR-7	Unit tests are highly encouraged and will be positively evaluated.
FR-8	Provide clear and complete documentation describing how to use, build, and test the solution.

Deliverables

1. Public GitHub repository named: **marker-coverage-<your_full-name>**
2. README.md containing:
    a. Problem overview & algorithm explanation (max 150 words)
    b. Setup & build instructions for Windows (MSVC / MinGW), macOS (clang), Linux (gcc/clang)
    c. Example command lines & outputs
    d. How to run unit tests and static analysis
3. (Optional) Dockerfile that builds & runs the app
4. (Optional) A simple pipeline diagram illustrating your detection process
5. Push all code and documentation to GitHub and send us the repository URL

Evaluation rubric (what we look for)

- Code quality – modern C++ idioms, clear structure, const-correctness, error handling, portability, comments.
- Algorithmic rigor & performance – correctness across provided and hidden test images, efficiency, and numerical stability.
- Build & tooling – clean CMake, multiplatform compilation, unit tests, CI/CD hints (GitHub Actions bonus).
- Documentation & clarity – concise README, useful inline docs, diagram.
- Git hygiene – logical commits, meaningful messages, .gitignore, license.

Submission timeline

- Time box: We would like to hear from you about your estimation. Please respond with your estimated delivery time no later than two days after receiving this assignment.
- Email the repository link (public or private invite) to sebastian@sodyo.com

Good luck – we're excited to review your solution!