

Imali AI receipt vision

An OpenCV 5 agent that decides whether a receipt photo can be trusted

TEAM	Ebenezer Tarubinga (Ebenworks)
PATH	Agentic Vision
LIVE DEMO	https://3-36-162-107.sslip.io
DATE	26 October 2026

Imali AI's agent keeps a small business's books from one WhatsApp chat, and owners send it photos of their receipts. This entry puts an OpenCV 5 step in front of that: it finds the slip in the photo, flattens it, scores how readable it is, cleans it and reads it with OpenCV's DNN text models, then an agent decides whether to record the expense, ask the owner to confirm the figure, or ask for a better photo. The rule that matters is that a total is recorded without asking only when a second line of the same receipt agrees with it. On 240 fresh made-up photos across eight capture conditions, it recorded 88% of the readable ones correctly without asking, against 78% for plain OCR, and it recorded no wrong total without asking, where plain OCR recorded 9.

01

1· Who it is for and why a photo is not enough

Imali AI is built for small shops: spaza shops, hardware stores, butcheries and market traders. They keep their trade in receipts and chats rather than in accounting software, and they can already send receipt photos to the agent, which writes each one into a double-entry ledger as an expense. The photos are taken on a counter with a phone in one hand. They come at an angle, under a hard shadow, in a dim shop, crumpled, blurred by a moving hand, washed out by a light, or with the bottom of the slip out of frame.

A reader that returns its best guess on every photo is the wrong tool for books. A wrong total that is recorded quietly is worse than no total, because the owner finds it weeks later, if at all, and it changes the profit, the tax figure and the cash count. So the problem is not only to read the total. It is to know when the reading can be trusted, and when it cannot, to say exactly what the owner should do: hold the phone still, move to better light, tilt the slip away from the lamp, or get the whole receipt in the frame.

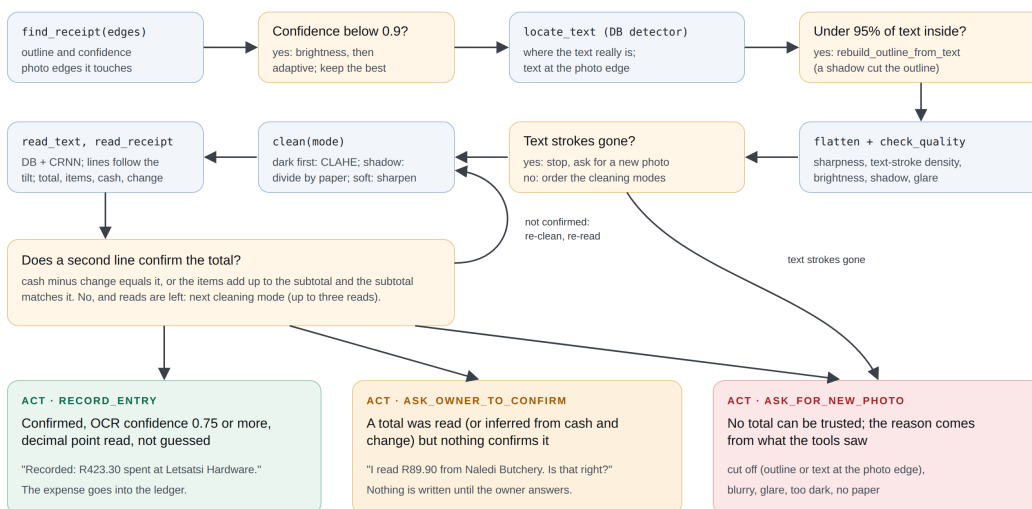
02

2· How the agent works

The agent runs a perceive, decide and act loop over a set of OpenCV 5 tools (figure 1). Every call is written to a trace with its arguments, its result, its time and the reason the agent gave for the next step; the demo page shows that trace with each answer.

The agent's loop: what OpenCV sees changes what happens next

Blue: perception tools (OpenCV 5). Amber: decisions. Green: actions. Every box is one entry in the trace returned with each answer.



Finding the paper. `find_receipt` has three methods: Canny edges on the L channel of $L^*a^*b^*$ with morphological closing and contour filling; Otsu thresholding of the blurred L

channel, since paper is usually the brightest large region; and CLAHE followed by a local threshold for dim scenes. Each returns a four-point outline from `approxPolyDP` on the convex hull (falling back to `minAreaRect`), a confidence from the area share and rectangularity, and the photo edges the outline touches. The agent tries the next method while the confidence is below 0.9 and keeps the best.

Checking the outline against the text. A hard shadow is the common way all three methods fail: the shadowed part of the paper looks like background, and the outline stops at the shadow line. So the agent runs the PP-OCrv3 DB text detector (`cv.dnn.TextDetectionModel_DB`) on a 960-pixel copy of the photo and measures what share of the text boxes lies inside the outline (`pointPolygonTest`). Under 95%, it rebuilds the outline as the four-sided hull of the old outline plus every text box. Text touching the top or bottom edge of the photo is kept as evidence that the slip is cut off.

Flattening, scoring and cleaning. `getPerspectiveTransform` and `warpPerspective` flatten the slip to 720 pixels wide. `check_quality` then measures the variance of the Laplacian, the density of strong Sobel gradients (text strokes), brightness, contrast, shading unevenness from a morphological background estimate, and the largest clipped patch after opening (glare). If the text strokes are gone, the agent stops and asks for a new photo. Otherwise the scores order three cleaning modes: dividing by the estimated paper colour to remove shadows, non-local-means denoising with CLAHE for dark photos, and an unsharp mask for soft ones.

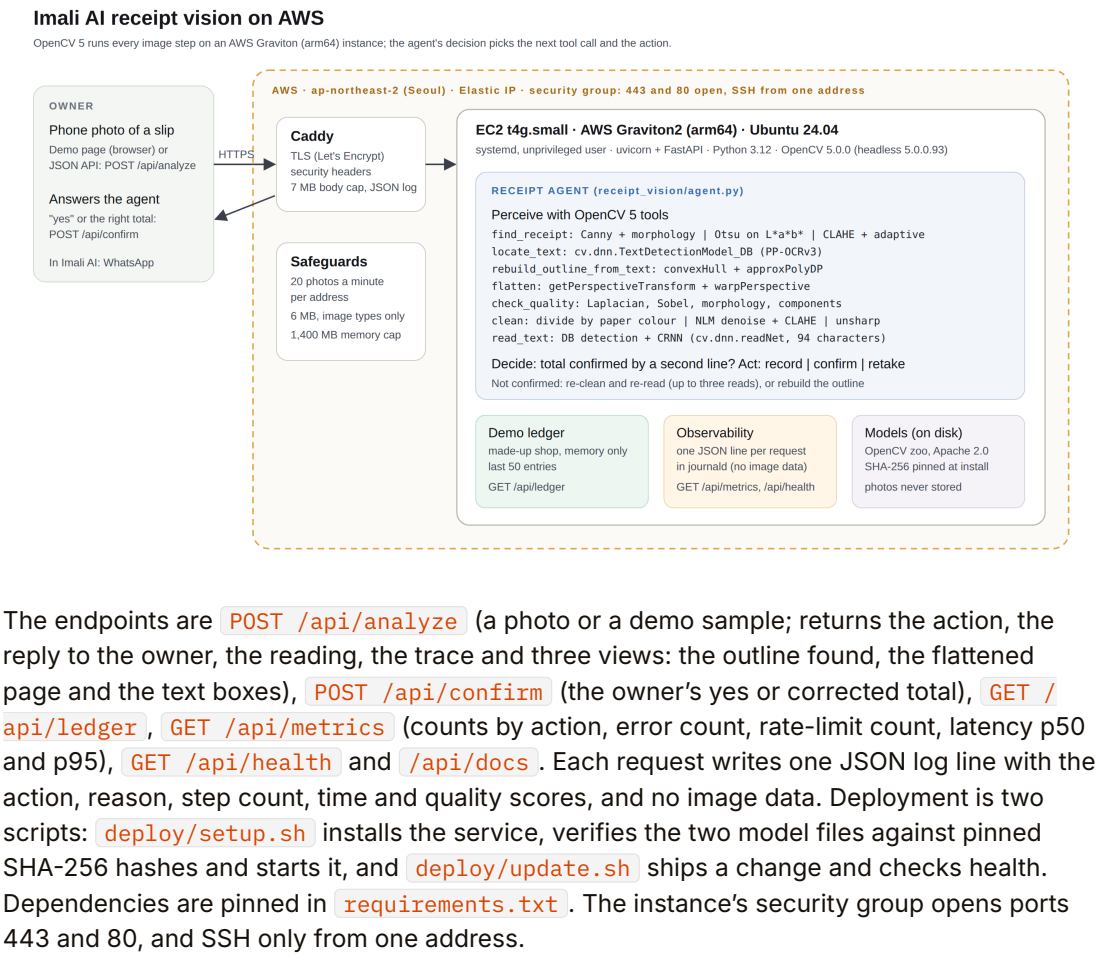
Reading. `read_text` detects text boxes on the cleaned page and reads each with the model zoo's CRNN (`cv.dnn.readNet`, a 94-character set of digits, Latin letters and punctuation, greedy CTC decoding). Boxes wider than nine times their height are read in parts, because the recogniser sees a fixed 100 by 32 window. `read_receipt` groups words into printed lines that follow the tilt of their first word, so a page still slightly keystoneed after flattening keeps prices on the same line as their labels. It finds the total line by fuzzy keyword match (TOTAL, TOTAAL, AMOUNT DUE and others, never SUBTOTAL), repairs common confusions inside amounts (O for 0, S for 5, a comma or dash for the decimal point), and reads the items, subtotal, VAT, cash and change.

Deciding. A total is confirmed when cash minus change equals it, or when the items add up to the subtotal and the subtotal matches it. If no reading so far is confirmed, the agent cleans the page another way and reads again, up to three reads. Then it acts. It records the expense only for a confirmed total read with confidence of at least 0.75 and a printed decimal point. It asks the owner to confirm when a total was read, or inferred from cash and change, but nothing confirms it; nothing is written until the owner answers yes or types the right figure. It asks for a new photo when no total can be trusted, and the reason comes from what the tools saw: an outline or text at the photo edge, missing text strokes, glare, darkness or no paper. The thresholds were set on calibration photos only (section 4).

03

3 · Running on AWS

The service runs on one AWS Graviton2 instance (`t4g.small`, arm64, 2 vCPU, 2 GB) in Seoul with an Elastic IP (figure 2). Caddy terminates TLS with a Let's Encrypt certificate, sets security headers, caps request bodies at 7 MB and writes a JSON access log. Behind it, a FastAPI app under uvicorn runs as an unprivileged systemd service with a read-only code directory and a 1,400 MB memory cap. OpenCV is `opencv-python-headless` 5.0.0.93 with two threads.



04

4 · How well it works

Each test photo comes from a generator that renders a receipt with a known total, places it on a counter-like background and applies one capture condition: clean, angled, hard shadow, low light, crumpled, motion blur, cut off or glare. Photos where blur, the frame edge or glare destroys the total are labelled unreadable; for those, the right answer is a request for a new photo or a correct total. The agent is compared with two baselines on the same photos: plain OCR on the raw photo that records any total it finds, and plain OCR with the agent's record, confirm and retake rules. A third line shows the agent with a single read, without the re-cleaning loop. All runs used the Graviton instance with two OpenCV threads.

The protocol had one break, and the numbers below handle it. Version 1 was frozen after calibration (seeds 1 and 11) and scored on seeds 7, 8 and 9. One failure in seed 7, a glare patch that wiped the same leading digit off the total, the cash and the change, led to a new rule: under measured glare the item column must also agree. Seed 7 was then no longer a clean test, so version 2, the deployed one, was scored on three new seeds (20, 21 and 22) that nothing was tuned on. Table 1 gives those results, as the mean and standard deviation across the three sets of 80.

MEASURE (240 PHOTOS, 197 READABLE)	PLAIN OCR, RECORDS ANY TOTAL	PLAIN OCR WITH THE AGENT'S RULES	AGENT, ONE READ	AGENT, FULL LOOP
Readable photos recorded right, without asking	78.2% ± 3.9	77.7% ± 3.3	86.8% ± 4.0	88.3% ± 2.5
Readable photos with the right total, recorded or proposed	78.2% ± 3.9	81.7% ± 0.3	88.8% ± 1.9	88.8% ± 1.9

MEASURE (240 PHOTOS, 197 READABLE)	PLAIN OCR, RECORDS ANY TOTAL	PLAIN OCR WITH THE AGENT'S RULES	AGENT, ONE READ	AGENT, FULL LOOP
Readable photos sent back for a new photo	18.8% ± 3.1	16.8% ± 1.6	9.1% ± 3.1	9.1% ± 3.1
Wrong total recorded without asking, all photos	9 of 240	0	0	0 of 240
Owner must type the total, all photos	0.0%	1.7%	1.7%	1.7%
Mean / 95th percentile time per photo	2.42 / 2.89 s	2.42 / 2.98 s	2.49 / 3.33 s	3.19 / 7.36 s

Table 1. Version 2 on fresh seeds 20 to 22. The exact 95% upper bound for 0 wrong totals in 240 photos is 1.5%.

The agent's gain over plain OCR comes from perception, not from the decision rules alone: the same rules on the raw photo lift safety (no wrong totals) but not coverage. Flattening and choosing a cleaning mode from the quality scores add ten points of hands-free recording. The loop changed course on 41 photos and rebuilt the outline from the text on 15. Version 1 on seeds 7 to 9 recorded 91.4% ± 0.9 of readable photos hands-free and recorded 2 wrong totals in 240, both on glare photos; that failure is what version 2's glare rule addresses, and version 2 recorded none.

CONDITION	PHOTOS	RECORDED RIGHT	OWNER CONFIRMS RIGHT TOTAL	OWNER TYPES	NEW PHOTO ASKED	WRONG RECORDED
Clean	48	48	0	0	0	0
Angled	63	63	0	0	0	0
Hard shadow	43	30	0	1	12	0
Low light	20	18	0	1	1	0
Crumpled	23	15	1	2	5	0
Motion blur	12	0	0	0	12	0
Cut off	13	0	1	0	12	0
Glare	18	13	0	0	5	0

Table 2. Version 2 by capture condition, pooled over seeds 20 to 22.

The failures are cautious ones. Hard shadow is the weakest condition: 12 of 43 shadow photos were sent back for a new photo. On 6 no total could be read; on the other 6 the stated reason was wrong (4 called blurry, 2 called cut off), because a dark shadow edge can pass for lost text strokes or for the frame edge. The owner gets a safe but unhelpful instruction there, and it is the first thing to fix. Every blurred photo and 12 of 13 cut-off photos were sent back with the right reason; the thirteenth was proposed to the owner with the right total. On the 4 photos where the owner would have to type the total, the agent read a figure but nothing confirmed it, so it asked rather than recorded. The full loop costs about 0.7 seconds a photo on average, and 7.4 seconds at the 95th percentile, when it reads three times. The code changes made after the evaluation hardened the web service (input caps, a bounded queue, per-browser ledgers) and capped the size of a flattened page at 4,320 pixels, which the generated slips (at most about 2:1) do not reach.

5 · Limits and responsible use

The test photos are synthetic. They were generated to cover the conditions seen in shops, and their answers are exact, but real slips bring faded thermal print, handwriting, stamps, other layouts and languages. The CRNN reader handles Latin script and the numbers on these slips; a slip in another script, or a total without two decimals, falls through to the owner. These are the reasons the agent records without asking only on arithmetic agreement, and leaves the rest to the owner, who always has the last word on what enters the books.

Photos are processed and dropped after each request; none is kept, and the logs carry no image content. Uploads are limited to 6 MB, 40 megapixels and 20 a minute per address, with one photo analysed at a time and at most four waiting. Each browser sees only its own demo ledger of up to 50 entries, held in memory; the demo never touches a real Imali AI business. Every receipt in the demo, the tests and the film was generated for this project, and the two models come from the OpenCV model zoo under Apache 2.0. The agent makes no decision about people: it reads a slip, and its one power is to record an expense that the slip itself confirms.

The next step is to place this step in front of Imali AI's vision model in production, so each photo is checked and flattened before the model reads it, and its total is cross-checked against the model's.