

Print/Scan Survival of PaperBack Data Pages

A fine-grain sweep of dot density, dot size, and scan factor

How well does a paper backup survive printing and scanning?

Data: Sep 28-29, 2026
Station: SCANNER01 - tech-center workstation
Chain: PaperBack 1.10 - rebuilt from source, automation-enabled
Print/Scan: Canon imageRUNNER ADVANCE DX CS840i / Epson Perfection V700 Photo

Automation: an AutoClaw (OpenClaw) agent carried out printing, scanning, decoding and measurement. Auto mode selected the models that did the sweep work (this revision of the report ran on GLM); the operator demonstrated the workflow by hand first, and the agent then drove it end to end.

Sweeps: pilot 01-15 | sweep 2 16-27 | sweep 3 28-42 | sweep 3c fine-grain 43-66
Sheets: 13 printed - pilot 3 | sweep 2 3 | sweep 3 3 | sweep 3c 4
Decodes: 665 automated attempts - every candidate at 6 scan factors x 2 scan variants

(sweep 2 was reprinted once after a station rebuild; counted once)

1 | Start here: what was tested and how to read this report

PaperBack is a paper-backup program: it encodes a computer file into a two-dimensional matrix of small printed dots - a 2D barcode carried on paper. To retrieve the file as stored in that 2D matrix, the page is scanned and decoded again. The catch: real printers and scanners are imperfect - fine dots print faint, scans blur, and paper moves. This project measured which dot settings survive the whole chain in practice.

We printed test sheets containing many candidate settings, scanned each candidate strip at several scan resolutions, and decoded every scan programmatically. The runs were driven through an automation interface added to PaperBack, with an AutoClaw (OpenClaw) agent supervising end to end - the operator first demonstrated the intended workflow by hand, and the agent then performed it automatically. The numbers match what a user would see in the application because the same code paths run.

The five knobs we swept

- Density - how many dot cells per inch the page uses (the dot raster, in dpi). Our main sweeps tested 50-125 dpi; the pilot went as fine as 1200 dpi. Higher density = more data per page, smaller dots.
- Dot size - how large each printed dot is drawn relative to its cell spacing (PaperBack's "Dot size" setting; our tests used 50-100%). Higher = larger dots, easier to recognize after printing.
- Scan factor - scan resolution as a multiple of density (we scanned every candidate at x2.0, x2.5, x3.0, x3.5, x4.0 and x5.0; e.g. an 80-dpi strip scanned at x3.0 = 240 dpi).
- Redundancy - how many spare blocks the page carries (held at 1:5: one recovery block per five data blocks, as in sweeps 2-3c).
- Compression - how the payload is stored (held at "Maximal" everywhere; affects page count only).

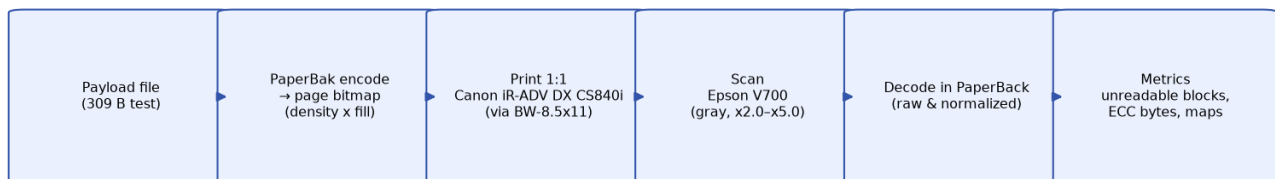
The two scan variants

- raw - the scanned pixels as-is (just orientation-corrected).
- normalized - a contrast-stretched version of the same scan. (Result: the raw bitmap always read first; the normalized variant is kept as a fallback.)

How to read the numbers

- Unreadable blocks (count) - data blocks the decoder could NOT recover. Unit: blocks. 0 = the page decoded completely. A block holds about 90 bytes of payload, so one unreadable block means about 90 bytes lost from that page.
- ECC corrections (bytes) - how many bytes needed error-correction inside blocks that DID decode. Unit: bytes per page. Lower = cleaner scan = more safety margin.
- "Reads from x3.0" - the smallest scan factor at which the raw variant decoded with 0 unreadable blocks.
- A sequence like "66 61 52 23 24 20" is the unreadable-block count at each scan factor in order: x2.0, x2.5, x3.0, x3.5, x4.0, x5.0 (66 blocks bad at x2.0 ... 20 bad at x5.0). "abrt" = the decoder could not even locate the page label (usually a too-coarse scan).

The paper-backup workflow tested in this report



The workflow: a file is encoded into dot pages, printed 1:1, scanned at multiple resolutions, and decoded programmatically for measurement.

2 | Equipment, constants, and the four sweeps

Print chain: sheets were printed 1:1 ("Actual size", best quality, monochrome) at a 2400x2400 dpi ticket through the BW-8.5x11 EnvisionWare virtual queue, which releases to a Canon imageRUNNER ADVANCE DX CS840i multifunction printer. Scan chain: Epson Perfection V700 Photo flatbed scanner, 8-bit grayscale, no sharpening. Decode: PaperBack 1.10, locally rebuilt from source with an added control interface so an automated agent could drive decode and encode without manual clicking (see the appendix for the patch set).



Canon iR-ADV DX CS840i



Epson Perfection V700 Photo

Canon and imageRUNNER ADVANCE are trademarks or registered trademarks of Canon Inc. Epson and Perfection are trademarks or registered trademarks of Seiko Epson Corporation. All other product names are the property of their respective owners.

Test payload: a 309-byte file, identical for every candidate; every strip encodes its own complete copy of it.

The four sweeps, chronologically

- Pilot sweep (candidates 01-15, Sep 28): tested very fine densities (1200 down to 160 dpi). Lesson: fine dots print faint and were not reliably decodable; also produced the ink-loss case studies.
- Sweep 2 (16-27, Sep 28-29): moved to coarse densities (160-50 dpi) and dot sizes 70-100%. First successes (60-80 dpi).
- Sweep 3 (28-42, Sep 29): mapped the density x dot-size grid at 50-120 dpi, filling the gaps between earlier points.
- Sweep 3c (43-66, Sep 29): fine-grain probes - seven candidates per winner zone (reliability, capacity, density edge) plus outlier fillers, all at one-pixel lever resolution.

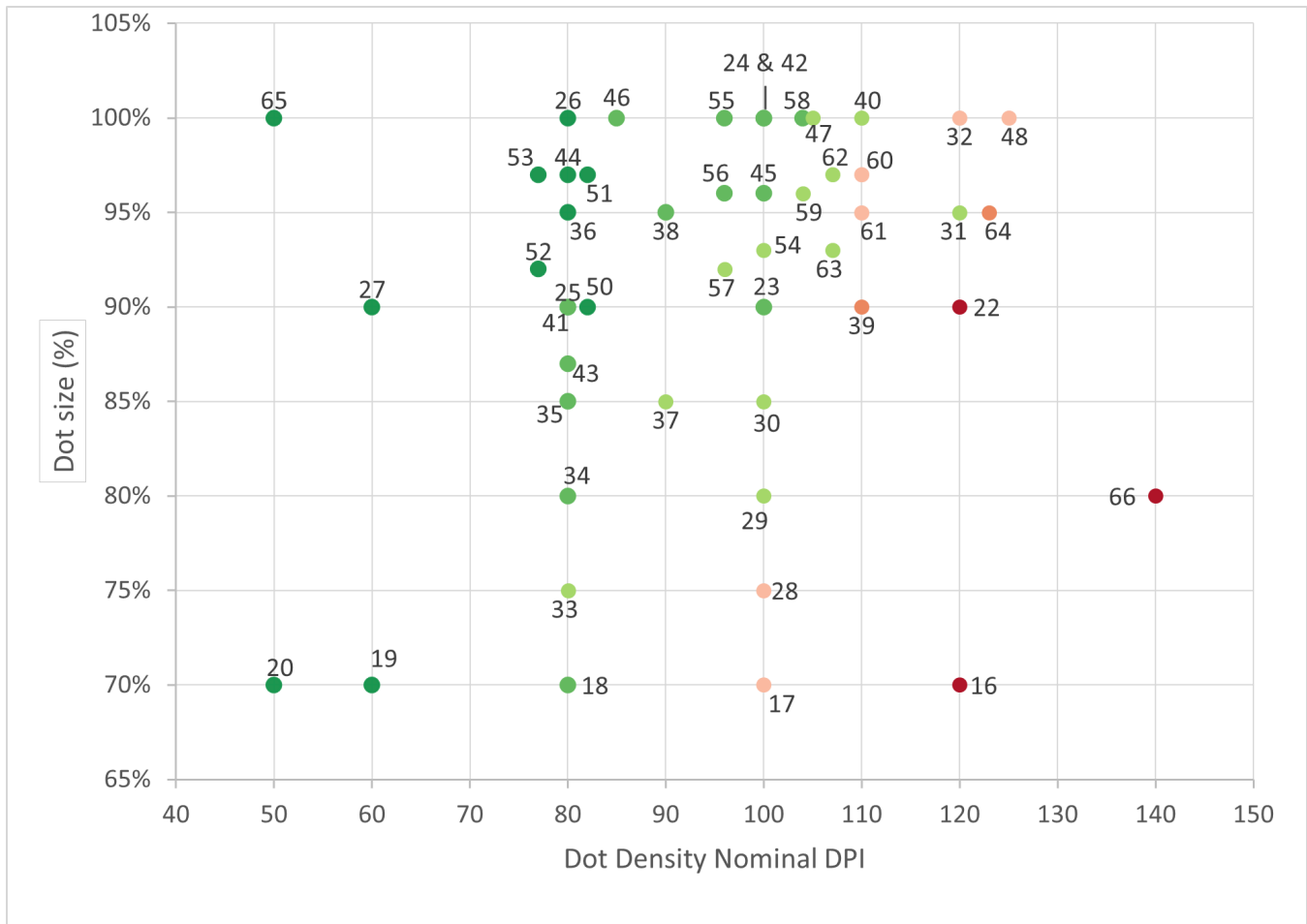
Every printed page was scanned at all six factors, in both variants, and decoded with the same pipeline the GUI uses - so the numbers match what a human would see in PaperBack, but are captured as machine-readable data (including per-block quality maps).

Fixed constants across all sweeps

Compression	Maximal (2/2)	(affects file size per page only)
Redundancy	1:5	(1 recovery block per 5 data blocks)
Encryption	off	Payload 309-byte test file
Header/border	on	Scan orientation handled automatically

3 | The big picture: which settings read, which fail

Each marker is one candidate setting: horizontal axis = dot density (dpi), vertical = dot size (%) - the two print settings from PaperBack's options dialog. Greens are clean reads (darker = fewer corrected bytes, i.e. more margin); reds are failures (lighter = more blocks still readable). Each marker is labeled with its candidate number. The amber class - reads only from the normalized bitmap - did not occur: the raw bitmap always read first.



Reading the shape: the frontier slopes up to the right - the finer the dot grid, the larger the dots must be (higher dot size) for the print/scan chain to survive. Above roughly 120 dpi even full dot size stops working (the "density ceiling").

4 | Master results - every candidate that reads cleanly

How to read this table

Capacity (blocks) = usable data-block cells in the candidate bitmap as generated and printed (one border column excluded; one block is about 90 payload bytes before 1:5 redundancy). Candidates were printed as bands on shared Letter sheets, so this is the capacity of one band, not of a full Letter page - verified directly against PaperBack's own grid (e.g. 80 dpi: 15 usable columns x 3 rows = 45 blocks; 100 dpi: 19x3 = 57; 120 dpi: 23x3 = 69).

Scan factor = scan resolution as a multiple of dot density (every candidate was scanned at x2.0, x2.5, x3.0, x3.5, x4.0 and x5.0; e.g. an 80-dpi strip scanned at x3.0 = 240 dpi). "Reads from" is the smallest of these at which the raw bitmap decoded with 0 unreadable blocks.

Best ECC = fewest corrected bytes seen at any clean factor (raw set). Lower = more margin.

ID	dens	size	cap	reads-from	best ECC	note
20	50	70	27	x2.0	4	
65	50	100	27	x2.0	0	bulletproof outlier
19	60	70	33	x2.0	18	first success (classic)
27	60	90	33	x2.5	23	
52	77	92	42	x2.5	9	
53	77	97	42	x2.5	12	
18	80	70	45	x2.5	140	
33	80	75	45	x2.5	197	
34	80	80	45	x2.5	67	
35	80	85	45	x2.5	76	
43	80	87	45	x2.5	43	
25	80	90	45	x2.5	9	sweep2 champion (90% dot size)
41	80	90	45	x2.5	64	
36	80	95	45	x2.0	23	
44	80	97	45	x2.5	6	record-low corrections
26	80	100	45	x2.5	25	
50	82	90	45	x2.5	29	
51	82	97	45	x2.5	7	
46	85	100	48	x2.5	40	
37	90	85	51	x2.5	295	
38	90	95	51	x3.0	100	
57	96	92	54	x2.5	219	
56	96	96	54	x3.0	114	
55	96	100	54	x2.5	82	
29	100	80	57	x3.5	171	
30	100	85	57	x3.5	542	
23	100	90	57	x3.5	120	
54	100	93	57	x3.5	238	
45	100	96	57	x2.5	49	Zone B efficiency pick
24	100	100	57	x2.5	61	
42	100	100	57	x2.5	84	
59	104	96	60	x3.5	250	
58	104	100	60	x3.0	86	Zone B capacity pick
47	105	100	63	x3.0	256	
63	107	93	63	x3.5	492	
62	107	97	63	x3.0	186	Zone C pick
40	110	100	66	x3.5	450	
31	120	95	69	x3.5	405	

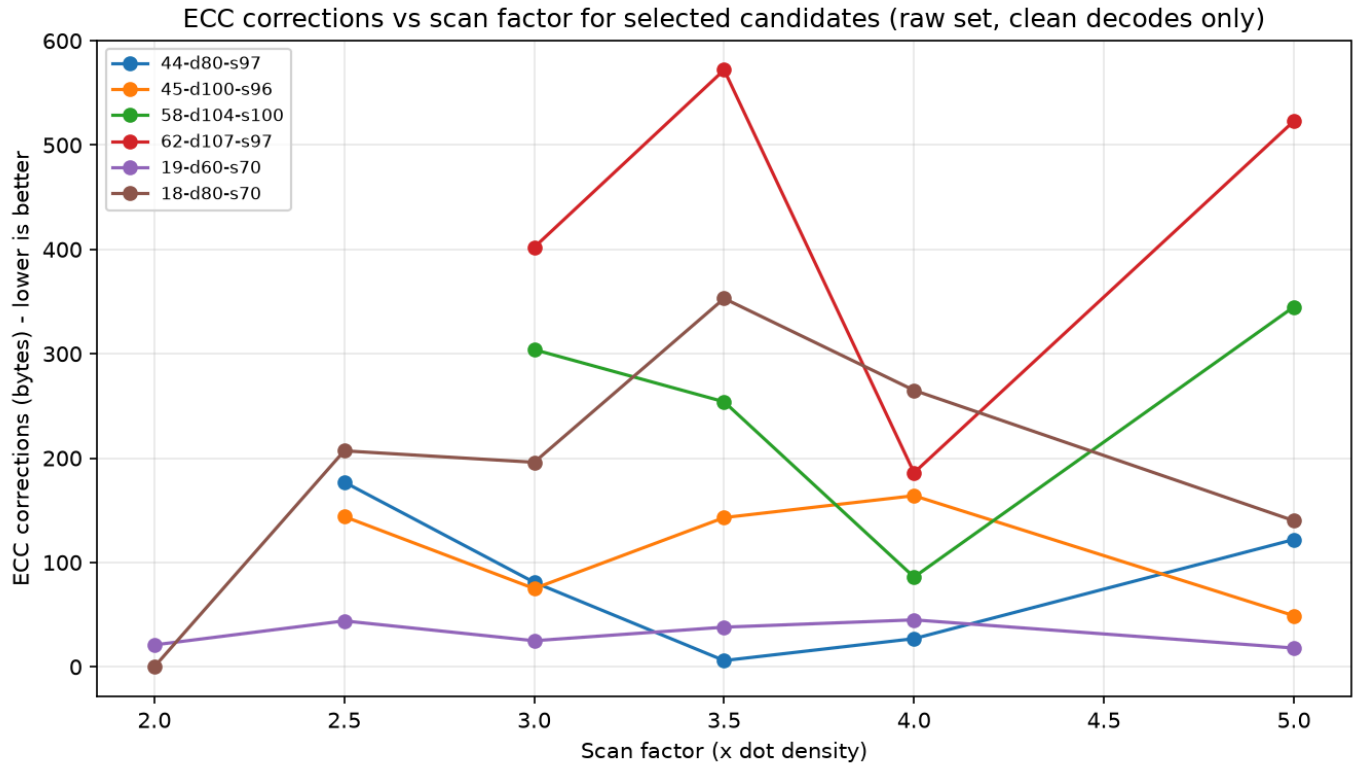
Candidates that never decoded cleanly (failures)

Even failed pages usually decoded part of their blocks - the "good at best" column shows readable blocks at the closest attempt, which is why some failures are near misses (1 block short) and others total.

ID	dens	size	good-at-best	closest attempt
16	120	70	0	69 unreadable at x2.0
17	100	70	42	15 unreadable at x3.5
21	160	70	0	92 unreadable at x3.0
22	120	90	0	69 unreadable at x2.0
28	100	75	39	18 unreadable at x3.5
32	120	100	68	1 unreadable at x3.5
39	110	90	9	57 unreadable at x3.5
48	125	100	23	49 unreadable at x3.5
60	110	97	65	1 unreadable at x3.5
61	110	95	46	20 unreadable at x5.0
64	123	95	2	70 unreadable at x4.0
66	140	80	0	81 unreadable at x3.0

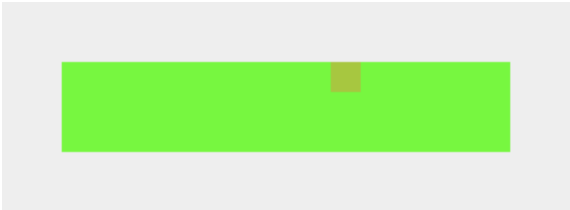
5 | Closer look: margins and failure modes

ECC corrections vs scan factor (lower = cleaner). The reliable candidates (44 , 45 , 58) sit low across factors; the density-edge candidate 62 carries more corrections and more scan-to-scan variation. Candidate 18 shows why "reads" and "margin" differ: it reads at every factor but carries several hundred bytes of corrections at low factors.

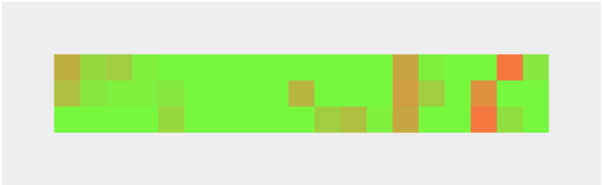


5.1 | Per-block quality maps - the decoder's own view

PaperBack tracks a quality value for every block on the page. The maps below use the application's own color ramp: yellow-green means the block decoded with no corrections, shading through orange as more bytes need correction, and black means unreadable; white/gray cells carry no frame (page border). These are the decoder's actual results, not simulations - eight candidates spanning the outcome range, from the record holder down to total failure.



44-d80-s97 at x3.5 - record margin: every block green (6 B corrected in total)



45-d100-s96 - efficiency pick: clean, light corrections only



18-d80-s70 - reads, but heavy ECC reliance (orange cells)



19-d60-s70 - classic 60 dpi: clean read, small corrections



60-d110-s97 - 1 block short at x3.5 (needs 100% dot size)



17-d100-s70 - near miss: most blocks readable, 15 short



66-d140-s80 - beyond the ceiling: no usable grid at all

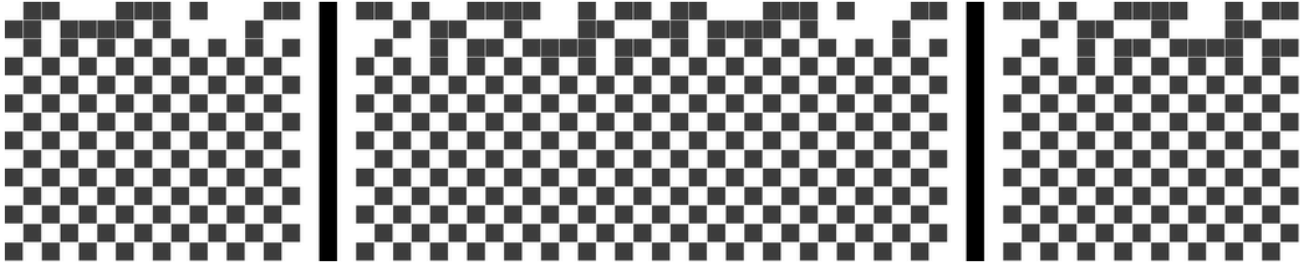


16-d120-s70 - 120 dpi at 70% dot size: every data block unreadable

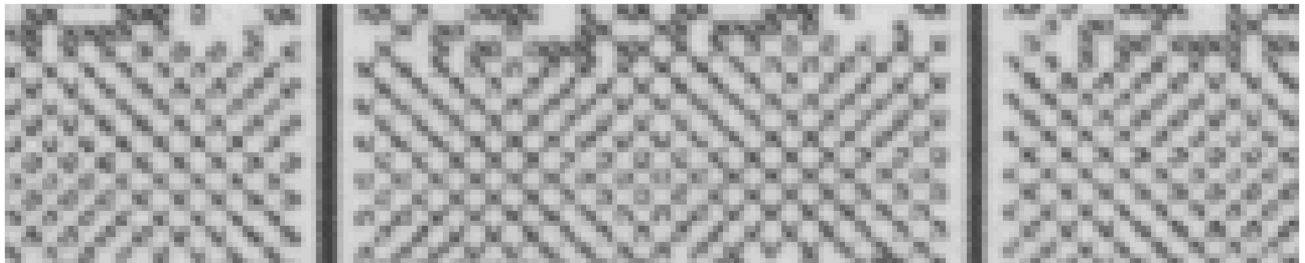
6 | Case illustrations - original vs printed-and-scanned, block-aligned

Each panel shows the same window from the crisp generated bitmap (top) and from the print/scan round trip (bottom), cropped on the dot grid so both halves line up dot for dot and share the same scale. Cases span the outcome range: two clean readers, one near miss, and three failure modes.

44-d80-s97 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



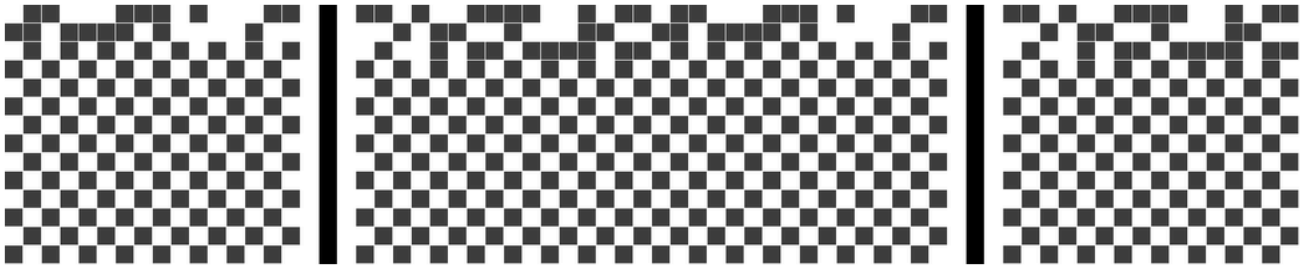
44-d80-s97 | PRINTED + SCANNED at x3.5 (280 dpi) - same area, same scale



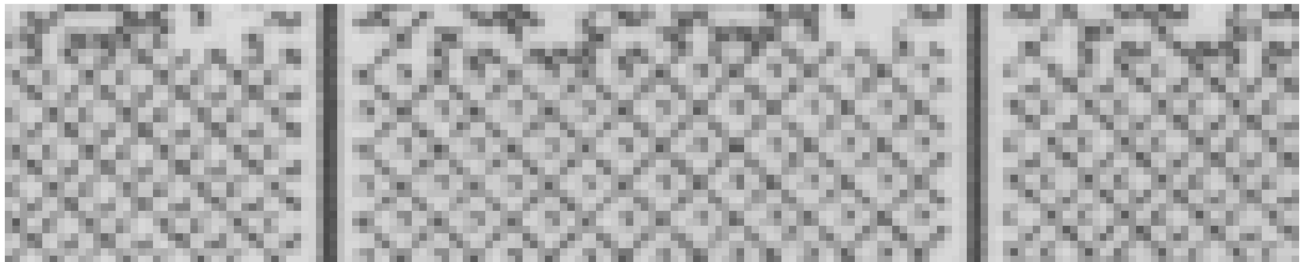
Reads cleanly: x3.5 scan, 6 bytes of ECC corrections across 45 blocks.

44-d80-s97 (record holder): dots survive the round trip almost intact - 6 bytes of corrections across 45 blocks at x3.5.

45-d100-s96 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



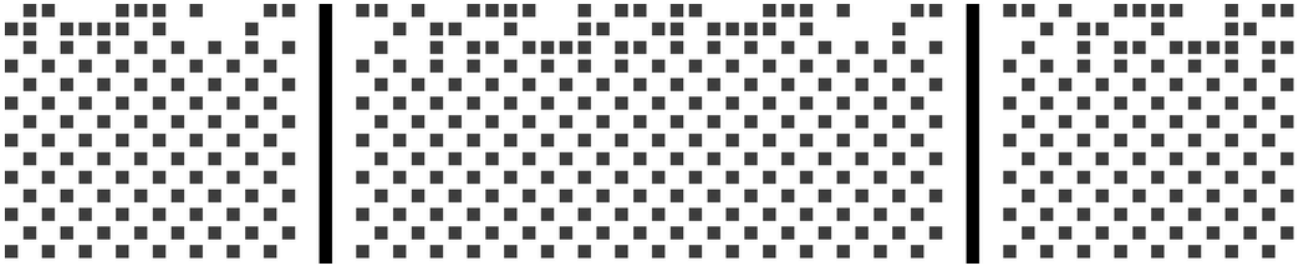
45-d100-s96 | PRINTED + SCANNED at x2.5 (250 dpi) - same area, same scale



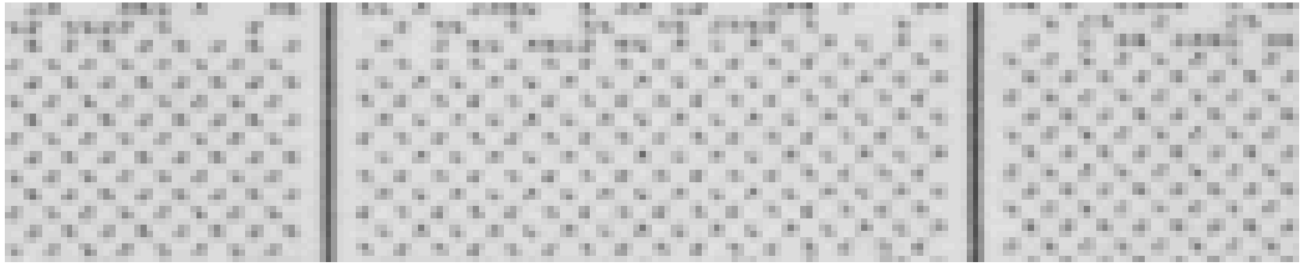
Efficiency pick: 57 blocks/page, reads from x2.5 with 49 B corrected at best.

45-d100-s96 (efficiency pick): 57 blocks, reads from x2.5.

19-d60-s70 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



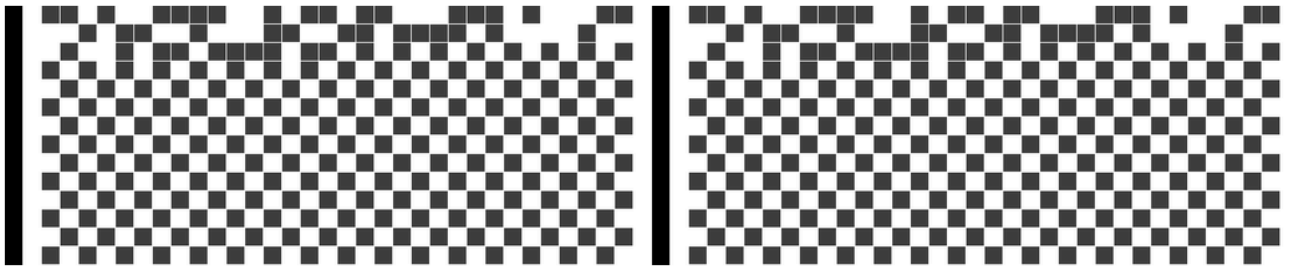
19-d60-s70 | PRINTED + SCANNED at x3.0 (180 dpi) - same area, same scale



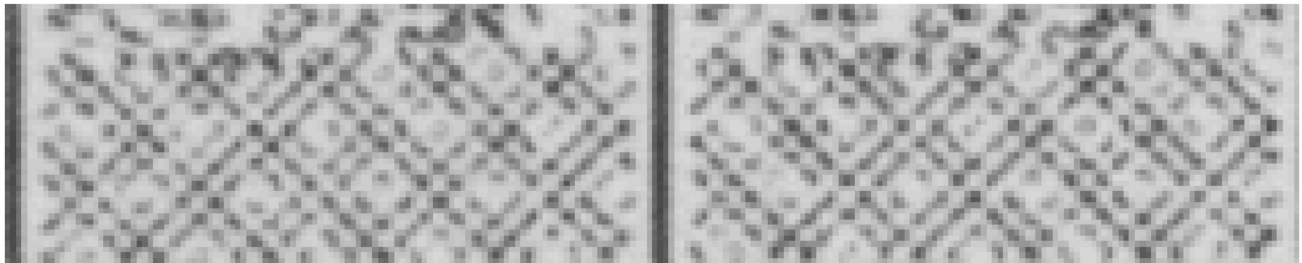
Reliable classic: 60 dpi reads from x2.5 with only 44 B of corrections.

19-d60-s70 (first success): large coarse dots remain solid after printing - readable from x2.5 with minimal corrections.

60-d110-s97 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



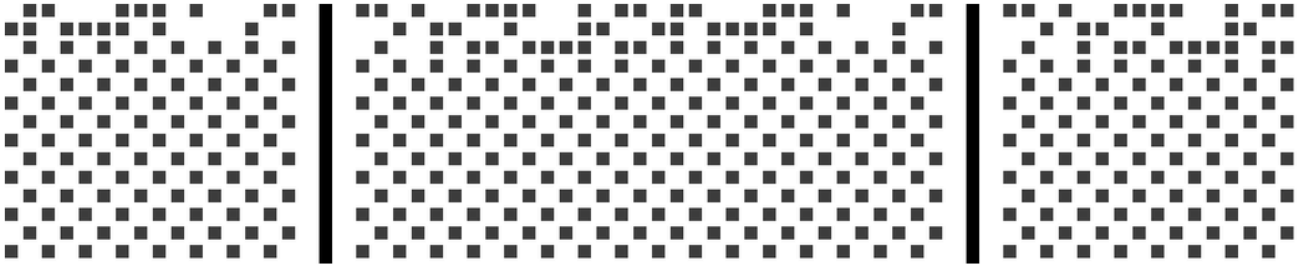
60-d110-s97 | PRINTED + SCANNED at x3.5 (385 dpi) - same area, same scale



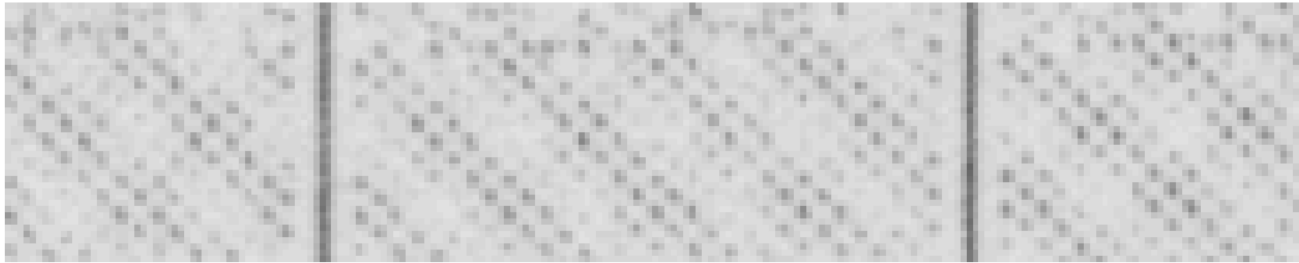
Near miss: 1 block short at x3.5 - 110 dpi needs 100% dot size to read.

60-d110-s97 (near miss): one block short at x3.5 - 110 dpi needs 100% dot size.

16-d120-s70 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



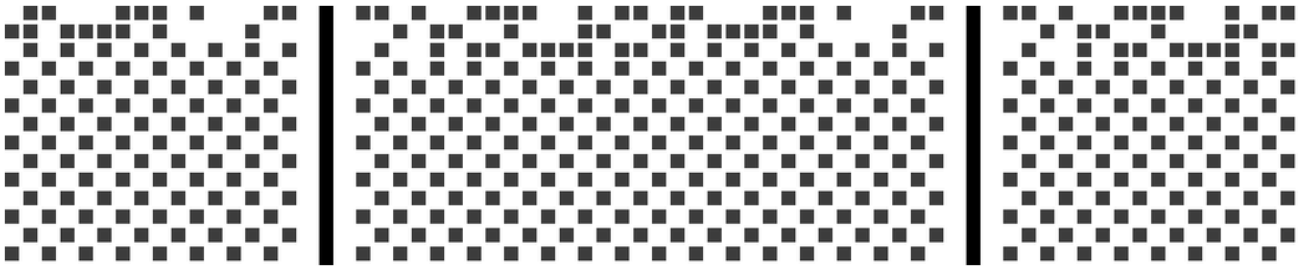
16-d120-s70 | PRINTED + SCANNED at x3.0 (360 dpi) - same area, same scale



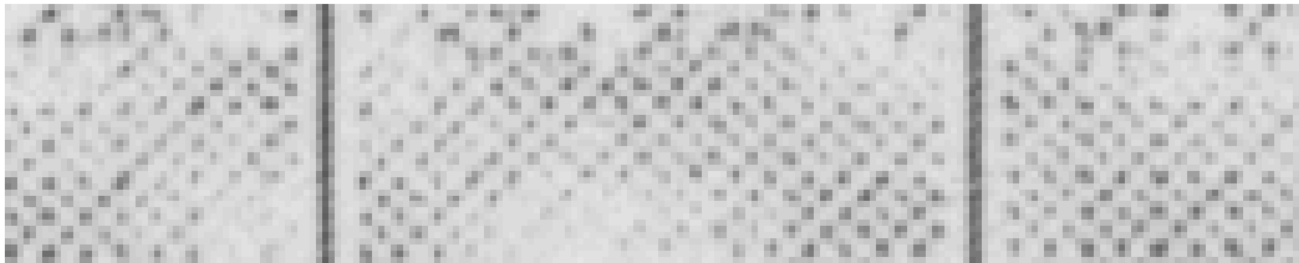
Fails at every scan factor: dots collapse to pale specks during printing (69 of 69 data blocks unreadable).

16-d120-s70 (failure): fine dots collapse to pale specks at print time - 69 of 69 data blocks unreadable at every factor.

66-d140-s80 | ORIGINAL generated bitmap - 70 x 14 dots, same area, same scale



66-d140-s80 | PRINTED + SCANNED at x3.0 (420 dpi) - same area, same scale



Never reads: 140 dpi dots exceed the survival limit entirely (density ceiling ~120 dpi at >=95% dot size).

66-d140-s80 (beyond the ceiling): 140 dpi dots exceed the survival limit entirely (ceiling ~120 dpi at >=95% dot size).

7 | Summary of findings

The dominant pattern is a readability frontier that slopes upward across the density / dot-size plane. Settings as fine as 120 dpi can read, but only when the dot size is at least 95%; the practical safe band across all sweeps is 77-107 dpi at 85-100% dot size. Dot size is the strongest lever at fine pitches: each additional 5% of dot size bought roughly half a scan factor of headroom (at 104 dpi, 96% dot size needed x3.5 while 100% dot size read at x3.0). Below the frontier, failure is gradual rather than binary - most failed pages still decoded part of their blocks, and several missed by a single block (60 at 110 dpi/97%, 32 at 120 dpi/100%).

There is a hard ceiling near 120 dpi. Candidates at 123, 125, 140 and 160 dpi failed at every dot size and scan factor tested, and the case illustrations show why: at those pitches the printer does not transfer complete dots, so the decoded grid simply vanishes. The pilot sweep measured the underlying physics directly - printing fine dots (≥ 160 dpi) loses on the order of 90% of their ink on this printer. In other words, the bottleneck is the printer's toner transfer, not the scanner or the decoder.

On the scanning side, three results stand out. First, the raw (un-normalized) bitmap always read first - contrast stretching never rescued a page that the raw scan could not read, and usually made things slightly worse, so normalization remains a fallback only. Second, there is a sampling sweet spot: x2.5 suffices for the coarser families (≤ 85 dpi), x3.0 for 100-107 dpi at high dot size, and x3.5 for the 110-120 dpi edge, while x2.0 is unreliable (page-label detection fails). Third, margin shrinks with density: typical corrections ran 10-80 bytes at 80 dpi, 100-500 at 100-107 dpi, and 400-700 at 110-120 dpi - "reads" and "reads with headroom" are different bars, and the ECC curves in section 5 make that visible.

Capacity scales with density as expected (data blocks per candidate band: 27 at 50 dpi up to 69 at 120 dpi, verified against PaperBack's own grid). The verdicts are repeatable: control candidates 41 and 42 reprinted the sweep-2 champions and read identically, with correction counts varying by about ± 50 bytes run to run. Finally, recovery is accumulative - repeated scans of a damaged page filled each other's gaps, and a saved union decode matched the original 309-byte payload byte for byte in three independent verifications.

8 | Recommendations

Reliability first (text you can always retrieve)

Use 80 dpi @ 97% dot size (candidate 44): it reads from x2.5, holds the record margin (6 bytes corrected), and carries 45 blocks per band. Alternatives: 77@92 (reads from x2.5, 42 blocks), 82@97 (7 bytes at x5). Scan at x3.0 minimum; x4-5 for strict recovery.

Capacity first (more bytes per sheet)

Use 100 dpi @ 96% (candidate 45: reads from x2.5, 57 blocks) or 104 dpi @ 100% (candidate 58: reads from x3.0, 60 blocks). Both need only half a scan factor more than the reliability pick for about 27% more data per band.

Density edge (maximum data per page)

Use 107 dpi @ 97% (candidate 62: reads from x3.0, 63 blocks) or, for the absolute maximum, 120 dpi @ 95% (candidate 31: reads from x3.5, 69 blocks) - but accept smaller margins (400-700 bytes of corrections; scan at x4-5).

Always-useful fallback

50 dpi @ 100% (candidate 65) decoded at every factor with virtually zero corrections - useful when a page must read, at the cost of capacity (27 blocks).

Scanning protocol that produced these numbers

1) Scan grayscale, no sharpening; 2) at least 3x the dot density (3.5-5x for the fine candidates); 3) decode the raw (un-normalized) bitmap first; 4) for pages that come up short, re-scan - partial results accumulate and can complete the file; 5) let automation verify each scan (unreadable blocks and ECC bytes are captured as JSON for every attempt).

Next steps

A production-style test with a realistic multi-page payload on the three picks; a redundancy probe (1:3 / 1:5 / 1:7) on the winners to trade capacity for damage tolerance; and, if desired, a fourth sweep around 110-120 dpi at 97-100% dot size to map the edge.

9 | Appendix - data & files

Relevant contents of Documents\PaperBak Sweepl:

sweep3c\ - fine-grain sheet, README, previews, results (p1-p4: CSV + full JSON)
sweep3\ - sweep 3 results (p1-p3)
sweep2-scan\ - sweep 2 scan results (dpisweep, dpisweep-p2, dpisweep-p3)
sweep2-scan\crops3x*-20260929\ - per-candidate raw/normalized scan strips
paperback-sweep-report-20260930.pdf - this report
report-data.json / report-figures\ - compiled dataset and figure sources
../paperbak-sweep3c-20260929.rar (etc.) - full scan archives for every sweep

PaperBack patches: the decoder used here is PaperBack 1.10 rebuilt from source with an automation control surface (named-pipe JSON interface, raw-value settings, quality-map export, and several GCC portability fixes). The complete patch kit, the build/patch/extend guide, and a patch-completeness audit live in E:\Documents\PaperBack Build\ (start with PAPERBACK-BUILD-GUIDE.md).

Method notes: results came from 665 automated decode responses (JSON). "Unreadable blocks" and "ECC corrections" are PaperBack's own counters, captured per decode; per-block quality maps are included in the JSON records. Scan factors were verified by reading the scanner's actual configured resolution back after each scan (never trusting request values).

Compiled 2026-09-30 15:59 | report generator: make_report.py (revised edition; original: pbreport-build.py / pbreport-pdf.py)