

Debug File Upload Instructions for **dvi3v** project

v3.9.2.208

Peter Felecan

with Claude (Anthropic AI), model Sonnet 5

October 12, 2026

Contents

1	Purpose	2
2	Quick Instructions	2
2.1	Enable Debug Mode	2
2.2	Files to Capture	2
3	What Each File Shows	3
3.1	debug_0_paper_detect.png	3
3.2	debug_1_binary.png	3
3.3	debug_2_cropped.png	3
3.4	debug_2b_step1_clean.png (–no-holes only)	4
3.5	debug_2b_step2_filled.png (–no-holes only)	4
3.6	debug_2b_step2b_despeckled.png (–no-holes only)	4
3.7	debug_3_final.pbm	4
4	Diagnostic Questions	5
5	How to Submit	5
6	System Information to Include	5
7	Privacy Note	6
8	What Happens Next	6
9	Conclusion	6

List of Figures

List of Tables

1 Purpose

When reporting issues with the `dvi3v` script, uploading debug images enables rapid diagnosis and fixes. This document explains which files to capture, what each one shows, and how to package and share them.

2 Quick Instructions

2.1 Enable Debug Mode

Run `dvi3v` with the `-d` or `--debug` flag:

```
dvi3v -d input.jpg output.svg
```

The debug output will show the temporary directory location:

```
Debug mode: temporary files in /tmp/dvi3v.XXXXXX
```

2.2 Files to Capture

From the debug directory shown above:

1. `debug_0_paper_detect.png` - reference paper detection result (present only if paper detection succeeded, rather than falling back to EXIF estimation or `--reference-mm`)
2. `debug_1_binary.png` - initial binary threshold image
3. `debug_2_cropped.png` - cropped object region
4. With `--no-holes` only:
 - `debug_2b_step1_clean.png` - edges cleaned
 - `debug_2b_step2_filled.png` - holes filled
 - `debug_2b_step2b_despeckled.png` - stray noise pixels removed, before the margin is added
5. `debug_3_final.pbm` - final bitmap before vectorization

6. `debug_info.txt` - a guided walkthrough of the above, generated alongside them

Important: Also include the exact command line and the input image.

Rather than attaching files one by one, bundle the whole debug directory (or the relevant `TestResults/` entry, if the case is also saved there) with `CreateTestDataArchive` — see [Data Exchange Workflow](#) for the archive tools and naming convention.

3 What Each File Shows

3.1 `debug__0__paper__detect.png`

Expected: The detected reference paper region WHITE, everything else BLACK, roughly matching the paper’s actual position in the photo

Purpose: Verifies which region was used to establish scale (px/mm), and by extension what the reported real-world dimensions are based on.

Common issues:

- Wrong region selected: a background area was mistaken for the paper (try `--paper-color` if the paper’s color wasn’t detected automatically)
- File absent: paper detection did not find a valid region at all (check whether an EXIF-estimated or `--reference-mm` scale was used instead)

3.2 `debug__1__binary.png`

Expected: Object appears WHITE, background appears BLACK

Purpose: Verifies threshold detection worked correctly.

Common issues:

- All black or all white: threshold value needs adjustment
- Multiple white regions: multiple objects detected
- Fragmented object: threshold too sensitive

3.3 `debug__2__cropped.png`

Expected: Just the object region, cropped from full image

Purpose: Verifies object localization and boundary detection.

Common issues:

- Too much background included: bounding box calculation error
- Object cut off: object detection missed parts
- Wrong region entirely: the reference paper (or another large, centered region) was picked instead of the intended object

3.4 debug_2b_step1_clean.png (–no-holes only)

Expected: Object edges cleaned via morphological operations

Purpose: Shows the state before hole-filling.

Common issues:

- Object degraded: cleaning was too aggressive for this object's fine detail

3.5 debug_2b_step2_filled.png (–no-holes only)

Expected: Object WHITE with internal holes filled solid

Purpose: Verifies the flood-fill hole-filling step.

Common issues:

- Holes still visible: flood-fill did not reach them (an enclosed hole should always be filled regardless of shape; if one remains, that is a bug worth reporting with this file attached)

3.6 debug_2b_step2b_despeckled.png (–no-holes only)

Expected: Same as the previous file, minus any isolated few-pixel noise specks

Purpose: Confirms stray noise was removed before the margin step would otherwise inflate it into a disproportionate, disconnected blob in the output.

Common issues:

- A small dot persists here (not just in the final output): the despeckle step did not catch it, worth reporting with this file

3.7 debug_3_final.pbm

Expected: Final binary image ready for vectorization

Purpose: Shows what potrace will convert to SVG.

Common issues:

- Excess margin: margin calculation incorrect
- Missing parts: earlier processing failed
- A separate, disconnected shape elsewhere in the frame: usually traces back to the reference paper, or noise, having been picked up as part of the object

4 Diagnostic Questions

If files cannot be uploaded, please answer instead:

1. What color is `debug_1_binary.png`?
 - All white, all black, or mixed black and white?
2. With `--no-holes`: what does `debug_2b_step2_filled.png` look like?
 - Are the object's holes filled solid, still visible, or is the whole image one uniform color?
3. Does `debug_2b_step1_clean.png` show a recognizable object?
4. What were the exact command-line options?
 - Especially: threshold (`-t`), auto-threshold (`-a`), color (`-c`), paper color (`-P`), `--no-holes`
5. What kind of object is being extracted, and on what color paper?
 - Dark object on light background, or vice versa?

5 How to Submit

1. Bundle the debug directory (input image, command line, and the files above) with `CreateTestDataArchive` — see [Data Exchange Workflow](#)
2. Share the resulting archive, and separately include:
 - System information (see below)
 - A short description of what was expected vs what happened

If the archive cannot be shared directly (e.g. it is too large for the channel in use), a file-sharing service works too — send the link along with the same information.

Bug reports, with or without an archive, go to Peter Felecan <pfelecan@gmail.com> (see also `man dvi3v`, BUGS).

6 System Information to Include

Run these commands and include the output:

```
dvi3v --version
convert --version | head -3
potrace --version
uname -a
```

7 Privacy Note

If the input image contains sensitive information:

- Crop or obscure sensitive regions before uploading
- Or create a similar test image reproducing the same issue
- Debug images are monochrome and abstract — usually safe on their own, but the input image itself may not be

8 What Happens Next

With debug files available, diagnosis can:

1. See exact pixel values and ImageMagick behavior
2. Test a proposed fix against the actual data that triggered the issue
3. Reproduce the issue locally
4. Confirm a targeted fix, rather than a guess

Without debug files, diagnosis is guesswork and may take much longer.

9 Conclusion

Debug files are invaluable for fixing issues quickly and definitively. The few minutes spent capturing and packaging them can save hours of back-and-forth diagnosis.

Thank you for helping improve `dvi3v`!