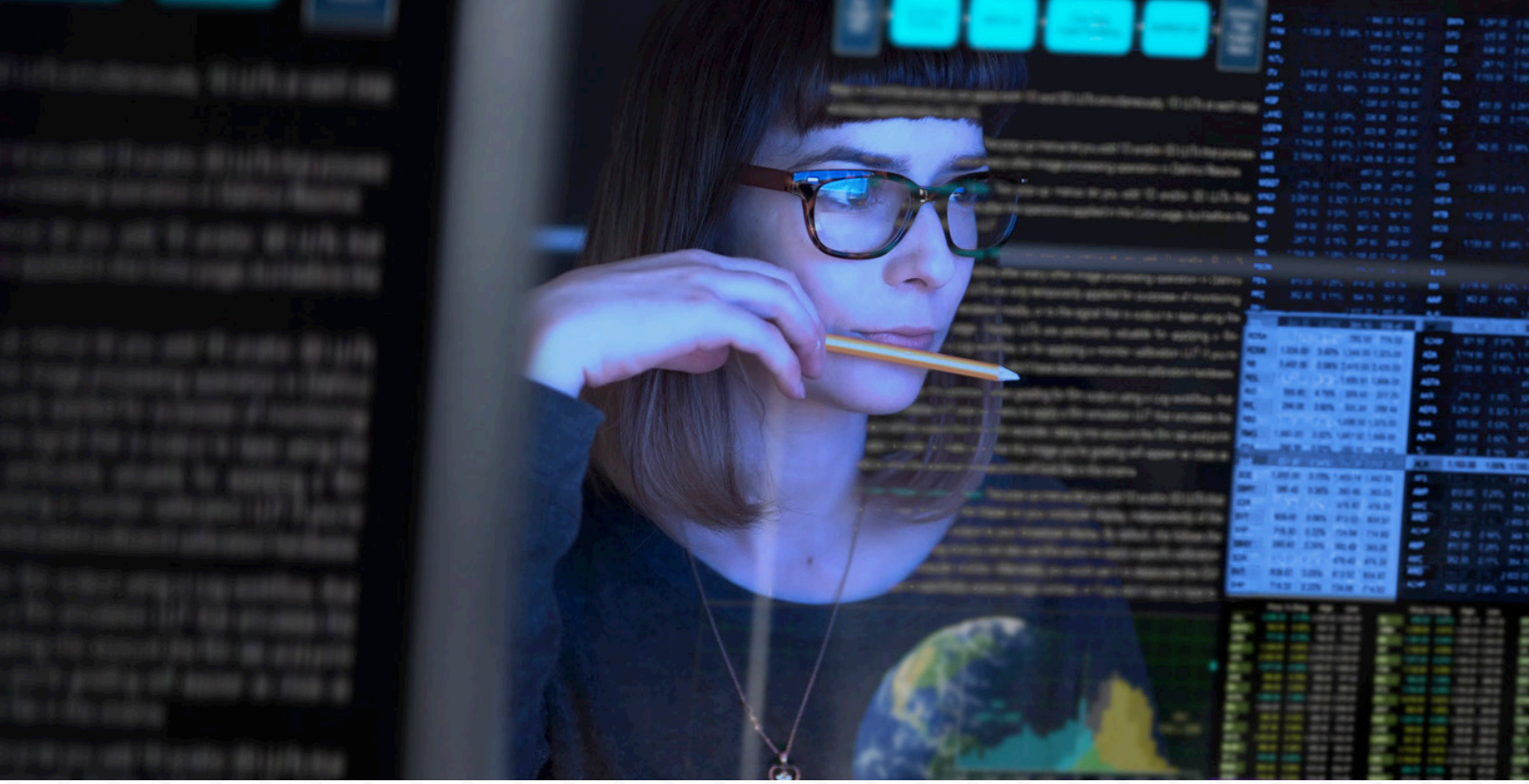
accusoft®

Visual Testing with Histograms

Dmitry Shubin
Software Engineer



Testing is a critical part of software development. On the other hand, massive testing done in an inefficient way can kill the development processes. This post discusses some traps in automated tests, and proposes a solution which works well in several of Accusoft's test suites.

Functional Testing and the Ground Truth Mousetrap

This post is mostly about **functional testing**, i.e. the tests which assure the product provides results the user wants to achieve, e.g. a document displays properly, a photograph has been cropped, a button is displayed on a web page, etc. This often means that result should look as expected, rather than just have specific text or parameters.

A simple approach to testing such cases is taking a screenshot of the working product and comparing it to rasterized "ground truth" (also referred to as "gold standard" or "reference data" - another screenshot that was taken when preparing the test and approved as "this is what we expect"):

Scenario: The viewer can remove sensitive content from the document

Given I have a document with sensitive content that matches pattern
When I upload the document into the viewer
Then the document shall display as groundTruth

This is often a bad thing.

Platform and Environment Dependency

Let's consider a case with an online document viewer application, which, among other things, can remove sensitive content when displaying documents. You want to ensure such content gets successfully removed.

With the raster ground truth approach, you would load the document into the viewer, take a screenshot of the browser window, and save this as the ground truth.

You later come to realize that you support three operating systems and four browsers. The screenshots are just slightly different but you don't want to lower comparison precision so you have to generate twelve nearly identical screenshots. After following this approach for some time, the test repository grows enormously and it takes a while to only download it to the test machine.

Dependency on Irrelevant Factors

Imagine you have a few hundred ground truth images, and one of the browsers gets an update, so its client window becomes a couple pixels taller. Boom! Now, you have to re-generate all ground truth for that browser even though nothing is wrong with your web page. Or maybe something IS wrong, and functionality got broken because of the new window dimensions? Oh no, you don't just need to re-generate ground truth, you need to review all of it!

Orthogonality

Functional tests should be orthogonal. If a test fails, it should tell what exactly is wrong with the product, rather than just "something is wrong." For example, if text layout changes, comparison with raster ground truth will fail, although the content removal feature works fine. It would be much better for page layout tests to fail instead, and content removal tests to stay successful.

Need for Manual Validation and Investigation

In order to approve a screenshot as a "ground truth," a team member has to review it with their own eyes. Similarly, when a test which uses ground truth fails, you have to visually investigate what exactly went wrong.

Reviewing a single screenshot is not a big deal, but multiplying this by the number of test cases and the number of supported environments makes it really boring. Moreover, boring manual work introduces the risk of oversight.

Histogram Approach

This approach utilizes a simple idea: use colors to distinguish things you want to see in the result and things you don't want to see, and then check for presence of those colors in the screenshot.

For the sensitive content removal case, prepare a test document with "non-sensitive" content in green color and "sensitive" content with red color, like this:

KEEP REDACT

KEEP REDACT KEEP

REDACT KEEP

The test will take the histogram of the displayed document and ensure there is no red color there (so we redacted the sensitive content) and there is still a specific amount of green (so we did not redact non-sensitive content).

The test will take the histogram of the displayed document and ensure there is no red color there (so we redacted the sensitive content) and there is still a specific amount of green (so we did not redact non-sensitive content).

Scenario: The viewer can remove sensitive content from the document

Given I have a document with sensitive content that matches pattern
When I upload the document into the viewer
Then the histogram of the displayed document shall have no entries of sensitiveTextColor
And the histogram of the displayed document shall have at least nonSensitiveTextColorEntries entries of nonSensitiveTextColor

Platform and Environment Dependency

The number of green points can be slightly different depending on the environment, but we don't need to check for exact number here; it is safe to use a rough threshold. If a bug is introduced in the application and a non-sensitive word gets removed, the difference will be much greater than possible differences between platforms.

Now, we have just one scenario instead of one scenario and twelve screenshots in our case of three platforms and four browsers.

Dependency on Irrelevant Factors

This check is much more focused than ground truth comparison. As long as the desired text displays and non-desired does not, the test will be happy.

Orthogonality

The test can still fail due to some unrelated issues, e.g. if the document starts to display with incorrect zoom or with wrong colors. However, this is much less likely than with ground truth comparison.

Need for Manual Validation and Investigation

This test has two clear verification steps: for absence of sensitive context and for presence of non-sensitive context. Just looking at the test report gives a good insight on what went wrong.

Other Use Cases

Histogram approach is great when you need to ensure that something is not present in the displayed result. Here are some other cases when it can be useful:

- Geometric transformations, such as rotation, scale, crop, etc. Paint several areas of the input document with different colors, and then calculate histograms of specific areas in the result view to make sure the area is filled with expected color.
- Color transformations and corrections. Prepare a test document with a few representative colors, and test for presence of expected colors in corresponding areas.

- Support for specific document formats, especially when you just need to ensure the format is supported, rather than check each and every feature of the format.

Limitations

The histogram approach assumes that you can use specially crafted documents for the test. If you need to test something which has a fixed appearance, such as UI, it is hard to use the histogram. If the UI is customizable, though, and you can supply a test stylesheet, that might do the trick.

The histogram approach does not really work when the test needs to verify fidelity - i.e. when something is displayed exactly as expected. We usually put such tests to a separate test suite, which only checks fidelity and does not check for more specific conditions like functional tests do. We run such tests less frequently and generally use them as a last resort, for the case when other, more specific tests don't catch the failure.

Implementation Notes

Histogram calculation is a pretty basic feature supported by many graphic libraries. For example, in Python, Pillow.Image.getcolors() can be used.

Keep in mind to use a limited number of colors in the test document, to avoid generating big histograms. Usually, just a few distinct colors are sufficient. Note, even if the page contains some black text on white background, the histogram will contain all the shades of gray due to antialiasing, so the default getcolors setting of maxcolors=256 will most likely not be enough. Histogram size of 65536 colors works well in our test cases.

Summary

The use of histograms is a simple but informative way of visual testing, which in many cases, can be considered as a better alternative to using rasterized ground truth comparison.