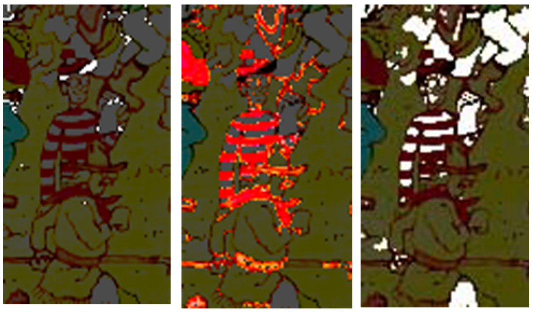
Jacob Sevart, UChicago Supplement to the Common Application

*I confess I don't know where Waldo really is, but I think I can still be of assistance: I can find him algorithmically. The following is a simplified, plain-English explanation of the procedure followed by the program I wrote for this essay. My source code is online at [https://github.com/jacobsevart/waldo_uchicago](https://github.com/jacobsevart/waldo_uchicago).*

Start with a high-contrast image of a page scanned from a Where's Waldo book. This image is a grid with a large but finite number of squares, each set to a single color.



First, process the image into something we can work with. Detecting Waldo will rely on his red and white striped shirt and his black hair, so find those colors:

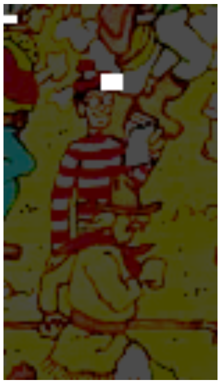
> Make a grid the same size as the original image. Write True in the boxes where the original image is red (or close enough to red) and False everywhere else. Do this again for white and black. *In the images on the right, True red and white squares, or pixels, expose the original. True black pixels are painted white, and False pixels are semi-opaque.*

Our first step in the search for Waldo's shirt is to find red pixels that immediately border white pixels above or below.

> Lay the grid of white pixels on top of the grid of red pixels, but shift the grid of white pixels down one space. Lay a new grid on top of these two and mark each space True if the two grid spaces under it are both True. Slide the grid of white pixels back to its original position, and then up one more space. Fill in True on your new grid if the spaces under it now are both labeled True. Label everything else False. *If it helps, think of these grids as overhead transparencies (though they're really just ideas).*

Now we need to find regions of red pixels that border white pixels which could potentially be components of sets of stripes.

> Draw rectangles circumscribed around contiguous regions of True on the grid of red-white border pixels. Give each rectangle a different number.
>
> If the rectangle is not big enough, erase it and set all the spaces inside it to False. *(This step saves unnecessary work and false positives down the road but is not strictly necessary.)*

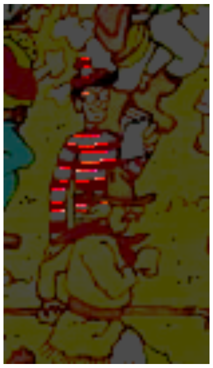


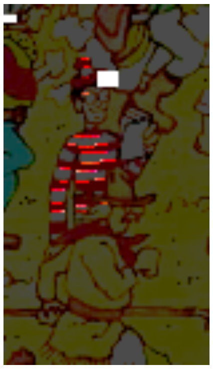
Do the same thing with the map that's True where the image is black, eliminating rectangles that aren't big enough to be part of Waldo's hair.

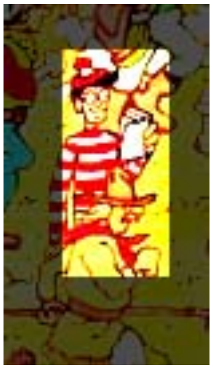
Next, we need to figure out which potential stripes make up a striped shirt. We'll say that this requires 3 rectangles vertically stacked with no more than 8 pixels between them. If a striped shirt also has a patch of black hair above it, we'll consider the whole thing a Waldo. *Sometimes, of course, it's not, but since this is a manhunt, we must avoid setting the filter too tight.*

For each red-white border rectangle that survived the filter:

> Place your finger on the bottom left corner of the starting rectangle. Go down one row, and then move your finger across that row until it is below the right edge of the rectangle. Go down another row and search from left to right. If your finger touches a rectangle, make note of the rectangle's number. Either way, keep going. Once you've moved more than 8 spaces down from the bottom of the last rectangle you found, stop.
>
> If you've written down less than 3 different rectangles' numbers for this starting point, it was not the top of a set of stripes, so evaluate another. If you *do* have at least three written down, this might be Waldo's shirt. Get out that grid of black pixels from earlier and check for hair:
>
> > Using the grid of black pixels, start from the coordinates of the first stripe rectangle and search an area 6 spaces wide by 30 spaces tall. If your finger touches a black rectangle, we'll consider it Waldo (it has something like a shirt and something like hair). Outline a large rectangle around him. If not, move on.







Of course, this simple, heuristic hack has limitations - on two pages of the book, it lights up a majority of the page. To reliably solve this sort of problem, I'm told, I'd want to create and train a neural network called a Botltzmann machine. So, UChicago, shall we revisit this question in CMSC 35400 (Machine Learning) or CMSC 35500 (Computer Vision)?

Disclaimer: I am not the first person to think of finding Waldo programmatically based on his stripes. I was inspired by http://stackoverflow.com/questions/8479058/how-do-i-find-waldo-with-mathematica. StackOverflow also helped me learn my way around NumPy, SciPy, Scikits-Image, and Python Imaging Library, four 3rd-party open-source libraries that do much of the heavy lifting. The *Where's Waldo* images used in this project are from Martin Handford's *Where's Waldo Now?*