COMP 150PP Class Exercise: Whose exam?  
(A real-world probabilistic-programming problem)  

October 8, 2014

This handout describes a real-world problem that, at least in principle, should be addressable using a probabilistic program. In class on Wednesday, we’ll look at possible solutions.

Students in COMP 170 mark their final exams by filling out a bubble form. The problem to be solved is to find out which student filled out which form. More precisely, your program is given

- A perfect image of a blank bubble form
- A list of all the UTLNs that could be coded for
- An image of a bubble form that a student has filled out

Both images are bitmap raster images: a rectangular array of pixels, each of which is either black or white.

The program should respond with a probability distribution over UTLNs from which we can see what UTLN is most likely to have been coded for, and what its probability is.

Your model should account for the following sources of uncertainty:

- Students sometimes make mistakes filling out the bubbles. They may fill out the wrong bubbles, or nearby bubbles. Students who realize their mistakes sometimes fill in the right bubble in addition to a wrong bubble!
- Bubbles may be filled in incompletely, and students may place marks outside the bubbles.
- The physics of the document feeder and the optics of the scanner usually result in an image that may be slightly translated, rotated, and scaled with respect to the original image.
- Both the scanning process and the printing process may introduce noise. As a simple model, assume that there is some probability of turning a white pixel black and that the same probability applies independently to each white pixel. Similarly for the black pixels, but with a different probability.
- Students or instructors may make other marks on the page; these marks will be seen, but they should be ignored.

Your task is to write a probabilistic program that will answer the question “given these UTLNs and these images, what is the probability distribution over UTLNs that were intended to be coded for?”

- Use the language of your choice. Haskell, ML, and $\lambda$ are all reasonable choices.
- At this early stage, don’t try to implement all the functions; just give their types. For example, it is enough to assume that you can scale an image about the origin, without necessarily knowing how the scaling works:

  ```haskell
  scale :: Positive Double -> Image -> P Image  
  -- returns the image obtained by scaling  
  -- the given image about the origin.  
  -- If in the result image, a pixel is 40%  
  -- black but 60% white, that pixel will be  
  -- represented by a suitable probability  
  -- distribution.
  ```

We’ll discuss results on Wednesday.