Congealing for Symbol Recognition in Digital Circuits

Jason Fennell and Joe Simons

May 2, 2008

1 Introduction

Sketches are a natural medium of communication used regularly in engineering, computer science, and many other fields. Whiteboards are an essential scientific tool, and Tablet PCs are becoming more and more popular. While sketches are excellent tools for communicating with other humans, they are limited by the inability of computers to recognize and analyze them. Many times sketches are used in the preliminary stages of design, such as for a software project or a digital circuit. However, there are no validation or simulation tools that can be used on a sketch. Computerized design programs, on the other hand, offer a multitude of tools for a designer to verify and interact with their design. Unfortunately, these design programs often have unwieldy and difficult interfaces for initial design and modification. Designers often spend more time fighting with an interface than they do productively creating. The long term goal of our research is to bridge this gap. We would like to create a program that is capable of recognizing unrestricted hand-drawn sketches on a Tablet PC and converting them into a format which a computer design program can analyze and interact with.

We call this problem sketch recognition, and it turns out that it is a very difficult problem with many sub-parts. We concentrate on the problem of Symbol Recognition in the domain of digital circuits. What we mean by this is that we only consider the recognition of sketches of digital circuits — images made of AND, OR, NOT, NAND, and NOR gates connected by wires — and we are only interested in recognizing the symbols in this domain. That is, given an unknown symbol we want to be able to classify it as AND,

OR, NOT, NAND, or NOR with the best possible accuracy. We call this Gate Recognition, and note that the particular problem of Gate Recognition does not have any solutions in the literature. There are a host of other problems in sketch recognition, and an interested reader can consult the Free-Sketch Recognition section of the reading list for Christine Alvarado's Pen Based Computing class: `http://www.cs.hmc.edu/courses/2008/spring/cs182-pbc/reading.html`.

2 Congealing: Motivation

The fundamental problem of Gate Recognition is to match an unknown gate as well as possible to one of the different classes of gates. We base our work off of the paper [1] which proposes the idea of congealing. This is basically a way of creating a "platonic gate" which will then be easier to match against an unknown gate.

We derive the term "platonic gate" from Plato's idea of forms. A form is the quintessentially idea or archetype behind an object. For example, according to Plato every horse in existence shares some piece or essence of horseness that allows it to be identified as a horse. This essence stems from each horse being a flawed version of the perfect horse archetype. When a new horse is born it is like it is stamped out from this archetype, but the stamping process introduces flaws that make that particular horse unique. The horse archetype is what can be called a "platonic horse" because it is the perfect horse, the horse that defines what it means to be a horse. What we want to do with congealing is find a platonic gate that defines what it means to be a particular logic gate.

In order to find a platonic gate, even conceptually, we must make some assumptions. First we assume that a platonic gate exists. Next, we use the idea that every real gate is created from this archetypical gate and assume that every gate that is drawn can be created from the platonic gate by some transform. A key assumption is that these transforms are invertible and have some sensible distribution to them. We have a set of drawn images $\mathcal{D}$ and we want to find a set of transforms $\mathcal{T}$ and a platonic image $\mathcal{I}$ such that for every $d \in \mathcal{D}$ there is a $t \in \mathcal{T}$ such that $t(d) = \mathcal{I}$, which is done by the process of congealing. Once we find $\mathcal{I}$ we can use it to improve classification by mapping an unknown image of a gate g into congealed space by some $t \in \mathcal{T}$ and comparing $t(g)$ to $\mathcal{I}$ by any one of a set of well-known methods

to determine if two images are similar. Essentially the process of congealing is one of noise-reduction. There is a mathematical justification of congealing in terms of probability in [1] for sufficiently interested readers. We highly recommend this paper for anyone trying to duplicate our work.

3 Methods

We want the computer to search for the platonic image for each of our gates. In order to do this, we reframe our question as an optimization problem. First, we read in a dataset containing many sketched examples of each type of gate. Then, for each type of gate, we convert the training gates to bitmaps. We map a colored pixel to one and a white pixel to zero. Then we create an `average image` as follows: for each pixel, we sum the value of the corresponding pixel in each gate in our training set, and divide this value by the number of gates. Thus the average image represents the probability of a given pixel being black in our training set. Our goal is to minimize the sum of `pixelwise entropy` of the average image, where the sum of pixelwise entropy is defined as

$$E = \sum_p H(v(p)).$$

$v(p)$ is the probability of a pixel p being black, as defined in the average image, and H is the discrete entropy function,

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p).$$

which looks like Figure 1 on the inverval [0,1]. Note that the minima of H are at 0 and 1, and the maximum of H is at .5. Thus, in order to minimize the entropy, we must be as certain as we can that a pixel is either white or black. In other words, if we imagine the probability of each pixel to be represented by a coin flip, we want to weight our coin as much as possible. We want, for each pixel, to be as certain as possible that that pixel is going to be black (or white). We do this by transforming our base images in a way that will increase this certainty.

3.1 Training

In order to minimize the entropy of an average image we want to apply transforms to the individual image in our dataset to make them each more

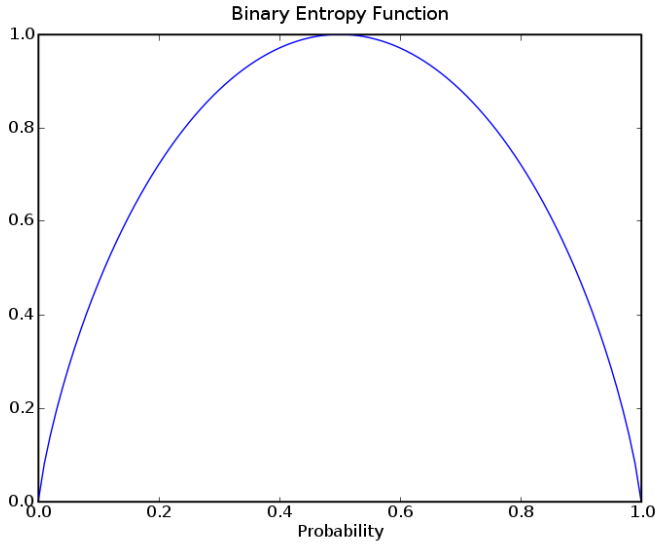

Figure 1: The binary entropy function H.

like the average. Mathematically this increasing likeness is represented by a decrease in total entropy. Visually it is shown by a increasingly sharp average image. For our training we restrict ourselves to only affine transforms, but other sets of transforms could easily be used.

The search algorithm that we use to minimize entropy is fairly simple. While decreases in total entropy continue happening we iterate through each image, and for each image we iterate through each possible transform in our set of transforms. We try that transform on that image and if total entropy decreases we keep that transform. After we have completed our iteration through all of the images we save the average image in whatever state it is for later use in classification. A little more formally in pseduocode, this is

```
// AvgImage before the warps are applied
AvgImage avgImg = new AvgImage(m_images);

// Stop training if improvements get small or number of iterations gets large
double entropyDelta = double.PositiveInfinity;
int numIters = 0;

while ((numIters < maxIters) && (entropyDelta > epsilon))
```

```
{
    for (int idx = 0; idx < numImgs; ++idx)
    {
        ImageTransform currentIT = m_images[idx];

        for(int warpIdx = 0; warpIdx < numWarps;  ++idx)
        {
            currentIT.apply(WARPS[warpIdx]);
            AvgImg newAvgImg = new AvgImg(m_images);

            if (newAvgImg.Entropy < avgImg.Entropy)
            {
                avgImg = newAvgImg;
            }
            else
            {
                currentIT.undoLastApply();
            }
        }
    }

    // Store the average image after this iteration
    avgImg = new AvgImage(m_images);
    models.Add(avgImg);

    updateTerminationParameters();
}
```

3.2 Classification

The problem of classification can be formalized as follows. We are given a universe with n different classes $c_1, \ldots, c_n$ and n platonic images that represent each of those classes $I_1, \ldots, I_n$. We are also given an image I of unknown class c. We want to find which class the image I is a member of with the most confidence possible. More formally, we want to find find i to maximize $P(c = c_i|I)$.

In our case we have $n = 5$ different classes — AND, OR, NOT, NAND,

and NOR — and want to find which gate an image is with the best possible accuracy. We do this with a simple distance based classifier. What this means is that we have defined a distance function $d(I, J)$ that gives us a measure of similarity between the images I and J (the distance is smaller the closer I and J are to being the same image). We take the distance between the unknown image I and the canonical images for each of our classes in congealed space and we designate I to be whatever class is closest to it. The point of the whole congealing process was to clean up the images to make this distance based comparison more accurate. The tricky part of this classification is that I must be in the congealed space of the class it is being compared to in order for an accurate distance to be taken.

However, through each step of training, we saved the average image. Now, in classification, we congeal the unknown gate to the sequence of average images for each different class. Thus at the first iteration of classification we compare the unknown gate to the average image saved in the first iteration of our training, and at the k^{th} iteration of classification we compare the unknown gate to the the average image saved in the k^{th} iteration of training. Hence we transform the candidate image into the congealed space, so it will be transformed to be more like the platonic version of itself, therefore reducing noise for our distance-based classification step. This gives us five different congealed images — I_{AND}, I_{OR}, etc. — which we can compare to the platonic images for each class with the distance function d. We assign I to be a member of the class that has its platonic image closest to the congealed version of I.

4 Results/Discussion

We have two types of results from this project. Our qualitative results demonstrate visually that the congealing process is working correctly and some of the problems that can arise in that process. Our quantitative results are not as encouraging, and show that we still need to do further work on our classification method.

4.1 Qualitative Results

We visually represent our average images as a bitmap, where the probability of each pixel being black is mapped to a corresponding gray-scale value. A

pixel that has a high probability of being black will be close to black in our average image, and likewise a pixel that has a low probability will be given a faint gray color. Thus, as the congealing process continues, the image appears to sharpen as the slightly misaligned gate symbols that make up the average image are transformed to better align with one another. Moreover, as the individual gates are better aligned, parts of the average image will also become darker and other parts will become white. Thus we visually confirm the fact that as we shift the individuals gates we are minimizing the entropy and therefore getting closer to finding the platonic version of the gate.

The next several pictures illustrate some of the qualitative successes and failures of the congealing process.

Figure 2: The average image for an AND gate before and after congealing.

Figure 3: The average image for an OR gate before and after congealing.

In Figure 2 we observe the improvement that the congealing processes has had on the average image for AND gates. In particular, the original average image for the AND gate (on the left) shows that there seem to be two different ways of drawing AND gates, on of which is more stretched out horizontally than the other. The congealing process merges these two ways of drawing AND gates and creates a much sharper and unambiguous image that we use as our platonic AND gate.

Figure 3 illustrates a similar process to Figure 2, but it is interesting to look at both figures together. The original average AND gate and the original average OR gate have enough noise in them that they are actually not that different. It is still possible for a human to distinguish between the two gates, but notice that the one place that the AND and OR differ, the straight versus curved backplane, is grayed out with uncertainty and variation in both of the average images. This means that it would be really hard to distinguish between AND and OR, and thus hard to take an unknown gate and establish if it is AND or OR by comparing it to these average images. Now consider the congealed images on the right. These are much sharper all around and should be much more useful for distinguishing between AND and OR, illustrating the usefulness of congealing.

Figure 4: The average image for an NOT gate before and part of the way through congealing.

Figure 4 is a negative example of congealing. This is what can happen when things go awry. Notice that the original average image for the NOT gate is very noisy and contains to distinct orientations of NOT gates. When we try to congeal this image our search algorithm is not able to rotate one set

of NOT gates far enough to be on top of the other set. Instead it just congeals to the point where the borders of the two orientations overlap, which is one of the most unambiguous areas in the first image. The congealing process rotates and shrinks each NOT gate in sequence until all that is left is a very dark black spot in the middle of the image. A large part of the reason that this happens is due to the high degree of noise in the data, and the inability of our search over transformations to find the right transformations to congeal the NOT. We tweaked our algorithm so that it used smaller steps, and it prevented this shrinking to a single point and did improve the NOTs somewhat. However, we would like to avoid having to do such tweaking, which is why we would like to implement a more general search framework for congealing in the future that uses a less ad-hoc search algorithm for transforms.

4.2 Quantitative Results

Despite the encouraging nature of our qualitative results, our quantitative results are actually very disappointing. The final run of data that we did was on 64 by 64 bitmaps of gates that had had a gaussian blur applied to them to allow the uncertainty in the original average images to work better with the congealing process. Unfortunately for our congealing algorithm, this increased resolution and gaussian blur actually helped the distance-based classifier based on the non-congealed average images work much better than the classifier of congealed images. We are not exactly sure why this is, but we think that the gaussian blur smoothed out variability in the original images and made non-congealed classification easier, whereas the sharper congealed images no longer match very well with the blurred classification images. Our results are as follows

	Non-Congealed % correct	Congealed % correct
AND	94	31
OR	83	22
NAND	92	69
NOR	17	100
NOT	13	13

The results for NOR and NOT are less significant because there were not a large number of examples. It is plain to see that the non-congealed case worked much better for AND, OR, and NAND. We have a large amount of future work that we would like to do to investigate and fix this problem, as

we think that our noise reduction procedure does have merit.

5 Future Work

There are several extensions to our project that we look forward to implementing in the future. First, we would like to create a generic function that maps from affine transforms to total entropy. This will allow us to use a variety of numerical methods to minimize entropy. Currently we are only using a naive hill descent algorithm, and sometimes we are getting stuck in a local minimum. We had hoped to use more sophisticated methods such as simulated annealing, but were unable to find numerical libraries for C# and did not have time to write such algorithms ourselves.

We would also like to extend the congealing algorithm to do sub-part recognition of gates. Instead of recognizing full symbols we would have the recognizer instead recognize parts of symbols such as the bubble on NOT, NAND, and NOR, the backplane of AND, NOT, and NAND, the backarc of OR and NOR, etc. If we could recognize each of the subpieces of a gate we could then combine those results to uniquely identify which full gate was present. Even more importantly, we would be able to identify if there were either missing or extra strokes in the gate we were trying to recognize. This would help inform the grouping process of sketch recognition, which is another problem we are pursuing in separate research.

Additionally, we would like to examine alternative distance metrics for our distance based classifier. While we did try five different distance metrics used in the literature along with our own entropy-based distance metric, we feel that they were the main sticking point of our algorithm. Since gates are so similar to each other, even when congealed, we believe that we need a distance metric that focuses on differences between the platonic images and measures the distance of a new image to all the features that actually make a difference between the platonic images.

Finally, we would like to investigate more closely weighting the classification of unknown images by the probability of the transforms that were applied to them, based on the probability of the transforms that were use in training. This is something that is done in [1] and that we duplicate, but we think that a closer look at this method may yield better results.

6 Conclusion

Digital logic gates are particularly difficult to classify accurately because they have such similar forms and are only differentiated by small features that different individuals draw very differently. The concept of congealing is to synthesize the "perfect" platonic version of each of the different gate classes, with the hope that the platonic version of each gate will more clearly exhibit the features that differentiate it from the other classes. While we have shown that congealing has some problems, particularly with noisy data congealing strangely, we have also shown qualitatively that congealing does a very good job of creating a platonic image of a gate. While our quantitative results are discouraging, we think that deeper analysis of our classification mechanism will improve those results. Most of this paper was focused on noise reduction through congealing, so the actual classification methods could still use further development. We eagerly anticipate improved results with some improvements to our classifier.

We would like to thank Professors Alvarado and Yong for their assistance with this project.

References

[1] E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. pages 464–471.