Final Project

Jake Brawer

May 7, 2017

1 Background

Insufficient minimal language experience in young children has been shown to have a devastating, lasting impact on a child's ability to develop normal language and literacy. Deaf infants are especially at risk due to the circumstances of their birth; approximately 90% of deaf children in America are born to hearing, non-signing parents. Currently, no relatively simple, cost-efficient solution exists. One might expect that inundating children with linguistic input from a tv/computer would be a helpful measure. However its been demonstrated that infants (and to a lesser extent, adults) cannot passively learn from inert sources. Instead, infants (and to a lesser extent, adults as well) require the learning process to be an interactive and contingent activity between two or more embodied agents.

In lieu of a good solution, my collaborators and I are designing a joint robot and virtual agent system that can (hopefully) provide linguistic input to deaf children. The idea is that the robot and the virtual agent (Figure 1) will be able to play of each others' weaknesses. The robot given its embodied nature is a salient stimulus capable of capturing and directing the infants attention, but does not have the dexterity to sign. The virtual agent on the other hand by virtue of being two-dimensional is not very engaging, but is infinitely dexterous and capable of signing. In order for the system to be an engaging conversational partner,it must have a sophisticated enough perceptual system so as to allow it to react in a natural and contingent which will hopefully bootstrap learning. One interesting finding to come from the linguistic literature is that deaf may produce "manual babbling," which is paralinguistic rhythmic hand gestures in the 1-2hz range. Currently, I am trying to devise an algorithm that can 1) detect infant hands and 2) parse manual babbling from nonlinguistic hand movements. Here I will be focusing on the algorithm for 1), which is a nontrivial computer vision

problem. However I have been able to simplify this problem significantly by transforming it into a skin detection problem, which we can do if we assume that only the skin on the infant's face and hands are visible (an easily enforceable rule in our pilot experiments).

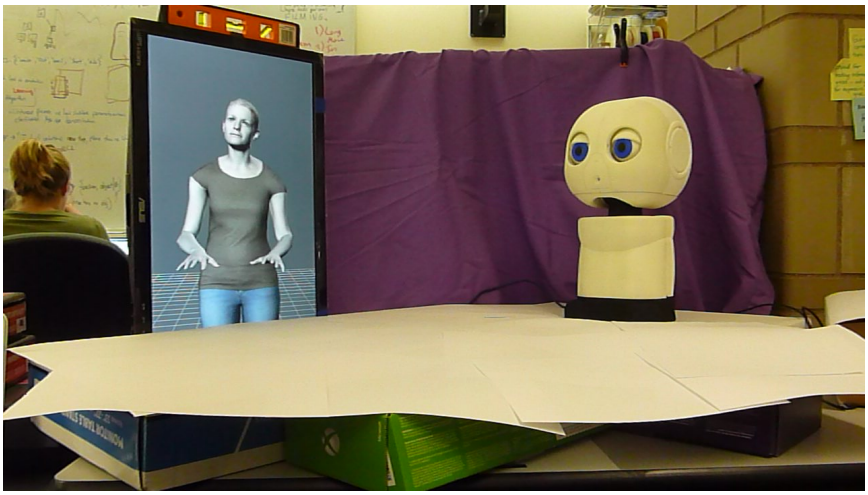

Figure 1: The robot-avatar system in action.

2 The Algorithm

A contiguous and uniform region in an image is called a *blob*. One common way to do blob detection is by convolving your mxn image with the difference the difference of two slightly different Gaussian distributions (DoG). The extrema of the resultant image roughly correspond to the centroids of the blobs. More formally, the DoG can be represented as:

$$\Gamma_{\sigma,K\sigma}(x,y) = I * (\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/(2\sigma^2)} - \frac{1}{2\pi K^2\sigma^2}e^{-(x^2+y^2)/(2K^2\sigma^2)})$$

Here σ is the variable of interest (generally for this sort of problem K is assumed to be ~1.6). In my original project proposal I had intended to focus on the optimization of sigma. The plan was to do something like this: I could first define the following functions:

$$r_j(\sigma) = Y_j - \phi(I_j, \sigma), \qquad j = 1, 2, ..., m$$

Here the function $\phi(I_j,\sigma)$ applies the DoG to an image I with a particular value of σ and spits out a 2x2 matrix of the coordinates of the two extrema. The 2x2 matrix Y_j of coordinates would be a subjective annotation of where I think centroids of the blobs are.

So the final equation I would have been optimizing would have been the following:

$$\underset{\sigma}{\text{minimize}} \quad r_1(\sigma)^2 + r_2(\sigma)^2 + ... + r_m(\sigma)^2$$

where m is the number of images I could be bothered to annotate. The problem with this though is that 1) I would have had to annotate s bunch of images by hand which is a mind-numbing task and 2) as I got deeper into coding this all up I realized there many variables that could be potentially optimized and that were also a little more important. This lead me to two realizations a) I could devise a metric that could automatically determine the efficacy of a changed variable with minimal intervention, and b) I could write a *general* gradient descent optimizer, i.e. one that computes the numerical gradient without me having to do the mathematical legwork required of close-formed solutions, and that could be applied to multiple functions. Without spoiling too much upfront, it turns out that b) was a terrible idea and I ended up wasting a lot of time on something that could have been easily avoided if I had done a little bit of math.

One of the difficulties with using DoG here is that even with the optimal σ for any given frame, there may be $n \geq 0$ extrema. The question, then, is how can you reliably and automatically determine which blobs belong to the baby's hand, if any? The method we employed involved creating a scoring function f that scored each detected blob based on a number of heuristics we thought denoted "hand-ness" and chose the two blobs with the highest left hand and right hand scores. In this case there were 9 features and they included things like how far a blob is from the predicted trajectories of the left and right hand blobs (this is easily calculated given you have the coordinates of the corresponding blobs from previous frames), how far the blobs were from the center of the image (generally this is not a great assumption but our experimental protocol enforces that the infant be at the center of the frame), etc.

One of the hypothetical advantages of this sort of this approach is that minimal calibration is required. That is, we would not need to go through the laborious processes of manually telling the system where the infant's hands are at initialization. Ideally, given the heuristics, the system should quickly zero in on the correct blobs. Of course, it's unclear off the bat how

much each of these factors should contribute to the final score, which is where optimization comes in handy. We can define a vector of coefficients $\alpha \in \mathbb{R}^9$ such that for any given frame we want to solve an optimization problem that looks something like:

$$\underset{\forall x \in X}{\text{argmax}} \quad \alpha^T f(x)$$

Where X is the set of all pixel coordinates of the detected blobs in a given frame. However, this is not enough; we need to optimize α and we need a way of judging whether are choice of α was good or not.

One thing we took advantage of is the fact that in a 30 frames-per-second (fps) video, an infant's hand is not going to be moving much from frame to frame. So the euclidean distance between which ever blob our scoring function chose at frame I_k and the blob chosen at I_{k-1} should be relatively small. Thus we have our optimization problem

$$\min_{\alpha} \quad \sum_{j=2}^{n} \quad \|(\underset{\forall x \in X_j}{\text{argmax}} \quad \alpha^T f(x)) - (\underset{\forall x \in X_{j-1}}{\text{argmax}} \quad \alpha^T f(x))\|^2$$

Where j is a frame in a video. From this formulation it's no entirely clear if this function is continuous, let alone differentiable. Had I realized this earlier I would not have spent so much time on this entire enterprise.

In any case what I did was write a method that approximates the numerical gradient of any function. This looked like:

$$\begin{aligned} \nabla g(x, y, \ldots, z)\Big|_{(x_0, y_0, \ldots, z_0)} &= \frac{g(x_0 + h, y_0, \ldots, z_0) - g(x_0 - h, y_0, \ldots, z_0)}{2h}\hat{x} + \ldots \\ &+ \frac{g(x_0, y_0, \ldots, z_0 + h) - g(x_0 - h, y_0, \ldots, z_0 - h)}{2h}\hat{z} \end{aligned}$$

Here the h I used was very small, approximately 1e-11 (Note: I made sure to use a double precision value to avoid accruing rounding errors assoicated with floating point operations). I implemented a pretty standard gradient descent algorithm with backtracking line search using g calculate the gradient. I used the following parameters $\alpha = 0,01$ and $\beta = 0.05$ for no other reason than they seemed good. Unsurprisingly, this algorithm did not work for the hand scoring function f. That is, the algorithm did not produce an appreciable difference in the coefficients. This makes sense, given the nature of the algorithm; In any given frame there are only finitely many points to choose from. This means that the domain of our distance function

Figure 2: Blob selection on filtered image using different coefficients. The image on the left uses the original coefficients derived by hand. The image on the right uses coefficients derived via grid search algorithm. Each grey circle denotes a blob. The bold grey circles denotes the blobs determined by the algorithm to be hands.

is highly discontinuous, and thus simple gradient descent is not appropriate in this instance.

3 The Algorithm: Redux

My gradient descent algorithm did not work for this particular problem (although I think the implementation is sound and should work for nicer functions), but I still needed a way to determine good coefficients for my scoring function. We are dealing with a function of sufficiently not-niceness so as to make brute-force methods the only viable strategy. Thus, I implemented a grid search algorithm to tune the coefficients of the scoring function. This basically involved exhaustively tweaking each coefficient one-by-one until a lower distance score is reached, and repeating this processes until some convergence criteria was reached.

In short, the algorithm worked well, but with surprising results. I tested all my algorithms on the same 10 second clip (so, a few thousand frames) of a baby taken from a previous pilot experiment. With the original coefficients derived by hand, the distance score was 6868. The coefficients derived via the grid search produced a score of 4692. It is difficult to quantitatively gauge the efficacy of these coefficients without annotating the data set, but at least qualitatively it seems like the grid search coefficients produced mixed/poorer results. Figure 2 is a side-by-side comparison of the same frame using the original and grid search-derived coefficients respectively. In general, the grid search-tuned scoring algorithm did not seem to detect the left hand as well

as the original algorithm. However, and remember this is my unsupported observations, it does seem like the grid search-tuned scoring algorithm did choose much more consistently, even if it was consistently wrong. That is, from frame to frame, it wasn't vacillating between the shoulder and hand blob, but rather was just sticking with the shoulder. In a way, this is actually a desirable feature, as a the end goal for this project is to detect infant hand gestures via frequency analysis of the blobs. Any sort of vacillation is bound to throw this type of analysis off.

While I did not get the results I wanted here, I believe these experiments put me on the right track. One thing to consider is that in subsequent infant trials, we are going to enforce that the infants and parents wear long sleeves. Under these conditions I believe our accuracy will go way up (not in Fig. 2: the skin on both faces were not being detected as blobs!). In any case it is clear the scoring function still needs work. I could, for example, penalize the blobs for being too close to the center of the image. Of course, though sometimes the infants hands will happen to be close to the center of the image, so we do not want to penalize too harshly.