# Understanding Probabilistic Inverse Graphics through Optical Digit Recognition with 2D Gaussians

**Ethan Chun (elchun@mit.edu)**
Massachusetts Institute of Technology

**Abstract**

Probabilistic inverse graphics has the potential to revolutionize visual inference in the face of uncertainty. However, the prerequisite knowledge to develop state of the art systems is immense. In this work, we simplify the issue and target optical digit recognition, one of the simplest inverse graphics problems known. While the task itself has largely been solved, we aim to provide a intuitive gateway into the world of probabilistic inverse graphics to ease development of more complex systems. To this end, we present a system to recognize and reconstruct the ten basic digits by modeling each as a collection of 2D Gaussians. Running inference on noisy point set images, we demonstrate a success rate of 0.95 over 50 trials per digit. Furthermore, we present two of our past attempts at this problem and provide intuitive reasoning on their strengths and pitfalls[1].

**Keywords:** probabilistic programming; character recognition

## Introduction

From robotic manipulation to real world navigation, state of the art visual systems are facing increasingly uncertain worlds. While traditional deep learning based approaches have shown great success in classic inverse graphics challenges, the discriminative nature of these systems makes them vulnerable to anomalies like occlusions, sensor noise, and dynamic scenes. Recent work in probabilistic inverse graphics (Zhou et al., 2023; Gothoskar et al., 2021) shows a promising direction forward, leveraging probabilistic inference to allow uncertainty quantification and more robust inference. However, the prerequisite knowledge and infrastructure to build these systems is immense and the insights deep. This work aims to lower the bar, providing a first entry into the world of probabilistic inverse graphics by building one of the simplest probabilistic inverse graphics pipelines. Rather than working on full scene segmentation, we focus on optical digit recognition and attempt to solve it with an intuitive and extensible method. As depicted in Figure 1, we present a method to recognize digits in a noisy environment by model each digit as a distribution of 2D Gaussians and running inference with Importance Sampling. We demonstrate the success of our method under assumptions about the base shape of the digits and propose extensions to generalize this method to richer datasets.

This project extends the `gen.dev` (Cusumano-Towner, Saad, Lew, & Mansinghka, 2019) introduction (standard and bottom up), data driven, and iterative inference tutorials with a focus on Gen's resources for defining novel distributions. While the final method may seem simple, we'd like the reader to understand alternative approaches that we used to attempt to solve the problem and likely reasons for their failure. We think this may be valuable for readers wishing to enter the field of probabilistic inverse graphics as it can highlight some of the common pitfalls in designing these systems. We include all three of our attempted methods in the "Methods" section, highlighting the lessons learned from each subsequent attempt.

[1]Code at is available at `https://github.com/elchun/gen_ocr`

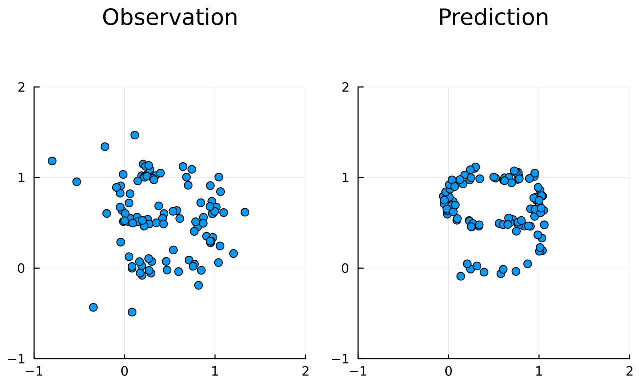


Figure 1: **Digit Recognition in Noise.** Given a noisy point set representing a digit, our model can infer what the digit was and create a denoised reconstruction.

## Related Work

### Classic Optical Digit Recognition

Before approaching this problem from a probabilistic lens, we refer to prior state of the art methods from classical computer vision and machine learning. A approach presented by Yann LeCun and colleagues in their seminal 1998 paper focuses on the problem by designing a convolutional neural network to recognize digits (Lecun, Bottou, Bengio, & Haffner, 1998). The network is modeled as a pyramid, working from the pixel level to the global. Furthermore, LeCun presents the now ubiquitous MNIST hand written digit dataset to train and evaluate his model. More recent architectures extend convolutional networks to work with more complex images by adding skip connections (He, Zhang, Ren, & Sun, 2015) or exclusively using attention mechanisms mirroring human cognition (Vaswani et al., 2017).

While these are purely data driven discriminative approaches, we take inspiration from these methods to better construct our probabilistic data generating process. In particular, we draw on the use of local and global features to help make our importance sampler more robust.

### Probabilistic Inverse Graphics

Aided by the development of dedicated probabilistic languages like Gen (Cusumano-Towner et al., 2019), probabilistic methods have increasingly been developed for use in inverse graphics pipelines. Notably, 3DP3 (Gothoskar et al., 2021) utilized a hierarchical scene graph to model objects non-parametrically, allowing for state of the art inference in the face of occlusions. More recently, 3D Neural Embedding Likelihoods (Zhou et al., 2023) combined features from deep learning with probabilistic methods to run 6D pose estimation. This method generalized well from sim to real and showed improved robustness in uncertain conditions.

While these methods perform well, the infrastructure and prerequisite knowledge required to develop these systems can be daunting. Therefore, we draw similar principles from these systems, but apply them to the much simpler domain of optical digit recognition.

## Methods

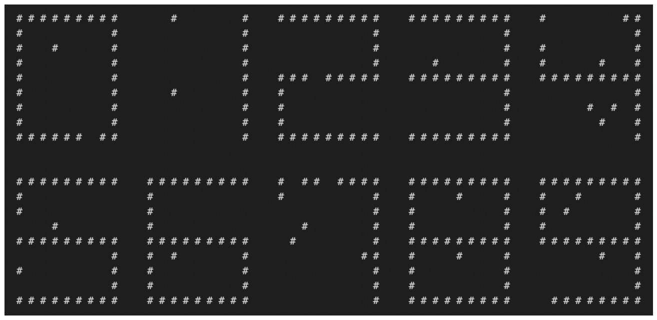
Our problem is defined as follows. Given a noisy "image" of a digit, determine what digit it is. Following the diversity of 3D data representations seen in inverse graphics, we allow this image to follow a traditional grid like layout or a more unconventional system like point sets.

One of the central advantages to probabilistic methods is the ability to encode prior information about a problem. In scene perception, one may include potential object geometry. In time series modeling, one may assume the target distribution follows some variant of a sinusoid. Consequently, a large amount of effort must be focused on how to encode this prior information. These methods focus on how we design a data generating process to encode these priors, and what the pitfalls of each design may be.

### Approach One: Inference on Pixels

**Data Generating Process** Our first approach looked to deep learning for inspiration. As such, we opt to represent our images as rectilinear images of size $9 \times 9$ binary pixels where digits are represented by the patterns of ones in these matrices. In classical computer vision, one might train a convolutional network to parse this structure.

To encode our priors, we create matrices representing archetypal numbers. Then, we assumed that the input images were generated in a two step process:



```
#########    #     #  #########  #########  #      ##
#       #          #          #          #          #
#  #    #          #          #          #  #       #
#       #          #          #    #     #  #    #  #
#       #          #  ### #####  #########  #########
#       #    #     #  #                  #          #
#       #          #  #                  #      # # #
#       #          #  #                  #       #  #
###### ##          #  #########  #########          #

#########  #########  # ## ####  #########  #########
#          #          #       #  #   #   #  #  #    #
#          #                  #  #       #  # #     #
   #       #            #     #  #       #  #       #
#########  #########   #      #  #########  #########
        #  # #     #         ##  #   #   #       #  #
#       #  #       #          #  #       #          #
        #  #       #          #  #       #          #
#########  #########          #  #########   ########
```

Figure 2: **Attempted Pixel Digits.** We attempt to render digits on a $9 \times 9$ grid with noise. However, note that many digits only differ by a small number of pixels, making inference difficult.

1. A digit is chosen from 0 to 9 with equal probability.

2. The digit is rendered by a pixel wise Bernoulli process where a pixel has a probability $p$ of being occupied if it is a one in the archetypal image, and a probability $1 - p$ of being occupied if it is a zero in the archetypal image.

When run alone, this generate noisy images of numbers as shown in Figure. 2.

**Issues** Unfortunately, while these digits look distinct to human observers, we struggled to use importance sampling to discern between the digits. While we omit a complete proof, we have strong reason to suspect that this trouble is with the way that importance sampling interacts with our proposal distribution.

Refer to digits "2" and "3" in Figure. 2. While obviously distinct to human observers, in pixel space the digits only differ on the three pixels in the lower vertical line segments. For an importance sampler to find a significant distinction between these two images, it would need to find that these variables were significant, then sample them in such a way that one digit is more likely than the other. This is particularly difficult with our pixel space renderer as each pixel is entirely independent from any other pixel in the rendered image. Therefore, it is difficult to determine which pixels are important to determining which number the image represents.

An interesting contrast can be made to deep learning. In a classic deep learning classifier, one may run the $9 \times 9$ pixel image through a Convolutional Neural Network (CNN) to extract features at various scales. These allows the network to find differences between the image classes at both the pixel level as well as a more abstract level. In contrast, our problem set up here only allows us to focus on the pixel level, making global inference difficult. Using these insights, we focus subsequent methods on building both small and large scale features into our data generating models.

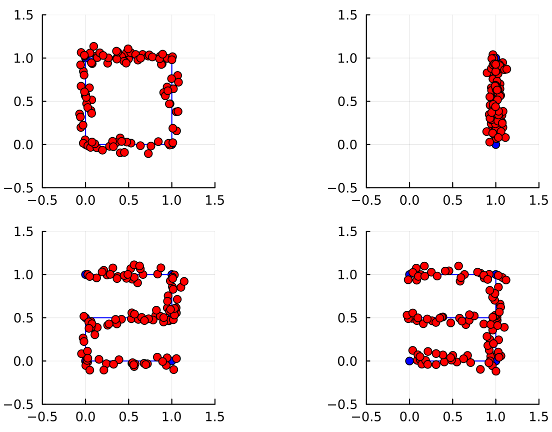


Figure 3: **Attempted Line Digits.** We attempted to render digits by sampling on along line segments in 2D space. While visually richer, this representation is not invariant to the order of the underlying point set.

## Approach Two: Parametric Lines

**Data Generating Process** Moving to the continuous domain, we attempt to mimic the digit layout seen in the pixel case, but provide a easier to optimize prior representation. To do so, we describe each digit as a collection of parametric lines. Then, to render a digit, we sample an arbitrary point on the line and add noise.

Specifically, we create each number by defining a piecewise parametric line where a parameter $t$ sweeps along the start to the end of this line in the range $t \in [0,1]$. The segments are not necessarily continuous but each $t$ value maps to a unique point. Then, we follow these steps to sample a number:

1. A digit is sampled from 0 to 9 with equal probability.
2. $k$ time steps $t_i \in [0,1]$ for $i \in \{1,2,...,k\}$ are sampled.
3. The location $\mu_i$ corresponding to each $t_i$ is calculated from the piecewise parametric line.
4. For each location, a final point is sampled from the normal distribution whose mean is $\mu_i$ and standard deviation is $\sigma = 0.05$.

As shown in Figure 3, these points form a point set image associated with the digit.

**Issues** We also found difficulty running importance sampling on this prior distribution, however, we believe that the underlying cause is subtly different than that of the pixel priors. Specifically, while it is now possible for importance sampling to find the render variables ($t_i$), the data generating process is not invariant to the order of the point set.

For example, take a point set of points $[p_1, p_2, p_3]$ at locations $[1,2,3]$. We can permute this set to $[p_2, p_1, p_3]$ and the resulting image will appear identical. However, the point sets $[1,2,3]$ and $[2,1,3]$ look nothing alike. Likewise, in our data generating process, the importance sampler must not just choose the correct distribution, but also choose the correct $t$ for each of the points in the point set. If the $t$s are permuted, as in our example, the estimated likelihood of the observed distribution given the hypothesized prior distribution will be extremely low. Therefore, our final attempt focuses on adding invariance to our continuous representation, thereby avoiding the issues of finding important variables and of finding a correct point ordering.

## Approach Three: Gaussians

Learning from the issues with the previous two attempts, we decide to continue using a point set representation of the data. However, given the trouble with parametric lines, we realize that any prior representation must be invariant to the ordering of the observed points. To this end, we draw inspiration from recent work on Gaussian Splatting (Kerbl, Kopanas, Leimkühler, & Drettakis, 2023), where a large number of 3D Gaussians are manipulated into representing a continuous 3D geometry. Our key insight is to model the data generating process of a point set image as sampling from a mixture of 2D Gaussians. Then, to represent an arbitrary digit, one can just encode the digit as the mixture of a finite number of Gaussians.

More concretely, we model the data generating process as a multi-step sequence. First, we create a mapping of each digit to a mixture of 2D Gaussians:

$$\mathcal{N}_d = \frac{1}{n} \sum_{i=0}^{n} \mathcal{N}(\mu_i, \Sigma_i)$$

For convenience, we follow the digit design from Approach Two and use each 2D Gaussian to model a line segment where the center of the line is $\mu_i$ and the covariance is $\Sigma_i$. Given a line of length $l$, the covariance matrix is diagonal with the variance along the length of the line equal to $\sigma^2_{long} = (l/2)^2$ and the variance along the width equal to $\sigma^2_{short} = 0.002$. This ensures that the 2 sigma confidence interval of the Gaussian covers the line.

Then for each sampled instance, we assume that

1. A digit is sampled from 0 to 9 with uniform probability.
2. A noise probability $p_{noise}$ is sampled from $Unif[0, 0.3]$
3. For each of the $k$ points to be rendered:

   With probability $1 - p_{noise}$, the point is rendered by sampling a point from the Gaussian mixture corresponding to the sampled digit.

   Otherwise, the point is sampled from a 2D normal distribution centered at $\mu = [0.5, 0.5]$ with diagonal variances of $[0.25, 0.25]$.

As shown in the observations of Figure 4, this produces digits similar in appearance to those in the line approach, however

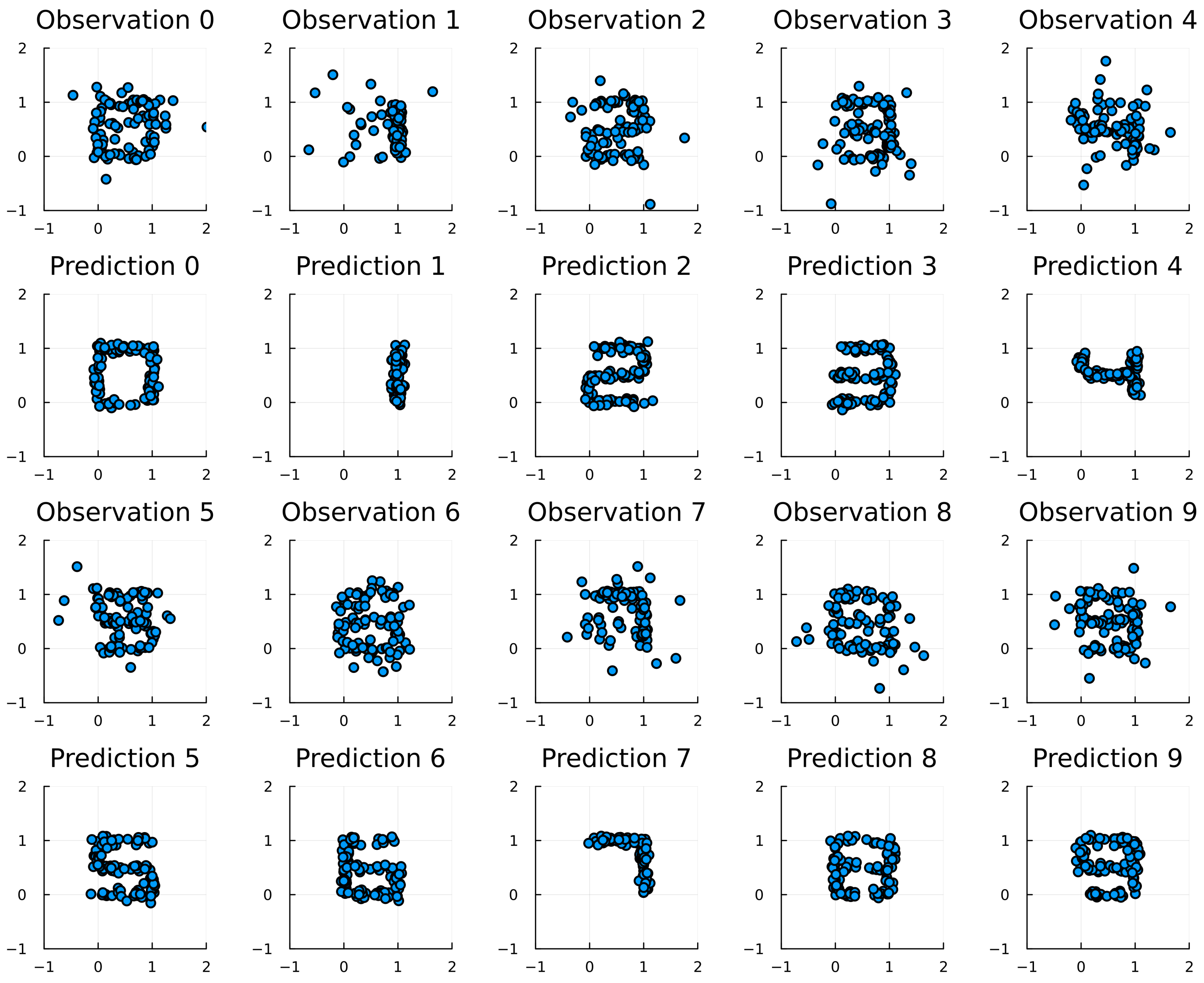


Figure 4: **Model Predictions of Digits.** Given noisy observations of digits, we infer the base digit and produce a denoised version of the observations. We represent each digit as a collection of 2D Gaussians, each parameterizing a line on the digit.

this representation has several key advantages over the previous methods. Firstly, the data generation process is invariant to the order of the observed points. Since we omit the intermediate variables $t_i$, there is no need for the importance sampler to predict the exact ordering of the observed points. Instead, its task is reduced to simply choosing the distribution that was most likely to have generated the observed data.

Furthermore, the continuous distribution avoids the confusion found in the pixel grid cases. Since we sample each point from a mixture of normals, we could immediately discern a rendered "2" from a prior representing "3", for example, since it would be extremely unlikely to have so many points on the vertical left of the image if the prior represented a "3". This increases the efficiency with which importance sampling operates and enables more robust predictions.

## Results

### Experiment Design

We design a simple experiment to evaluate our method. For each digit "0" to "9", we use our data generating procedure to sample the Gaussian representing the digit with $p_{noise} = 0.3$. Then, we use our importance sampler to infer which digit was selected. Finally, we render the inferred digit without noise. While this limits the interpretation of our results to processes with the exact same data generating procedure, we think this represents a promising preliminary result as neither of the other methods discussed could reconstruct a sample from the generating distribution.

### Experiment Results

As shown in Fig 4, our 2D Gaussian method is able to infer the identity of and reconstruct all of the digits from "0"

| Digit | Average success rate | Average lmls |
|---|---|---|
| 0 | 1.00 | -53.972 |
| 1 | 1.00 | 19.873 |
| 2 | 1.00 | -50.331 |
| 3 | 0.94 | -47.718 |
| 4 | 1.00 | -25.416 |
| 5 | 0.92 | -53.985 |
| 6 | 0.92 | -58.596 |
| 7 | 1.00 | -20.940 |
| 8 | 0.86 | -65.961 |
| 9 | 0.86 | -60.580 |
| total | 0.95 | - |

Table 1: **Success and LML.** Average success rate and inferred log marginal likelihood for digits 0-9 over 50 trials.

to "9". Furthermore, as shown in Table 1, across 50 trials for each digit, our model had an average success rate of 0.95 with a success rate of 1.0 for digits 0, 1, 2, 4, and 7. We also recorded average log marginal likelihood for each digit, finding averages in the range of -60 to -20 for all digits except "1" which has a average lml of around 20.

## Discussion

### Comparison between Methods

While each data generating process produced reasonable looking data, only the final 2D Gaussian Method allowed for quick and accurate inference. This highlights the importance of how the prior must be represented. In particular, we found that the prior must:

1. Be invariant to the order of the point set.
2. Be expressive enough to differentiate between each digit.
3. Provide means for importance sampling to pick up on important variables.

Given these constraints, it is likely that similarly constructed continuous priors will perform similarly to our 2D Gaussian system.

### Future Work

The 2D Gaussian priors can be easily extended to represent more expressive data. For example, to represent distorted or translated digits, one may add a sequence of parametric transforms to the Gaussian distributions. Then, one could use importance sampling or MCMC to infer both the base distribution and the transform's parameters.

It is also possible to evaluate this method on a larger dataset. Since we use a point set representation, one could easily convert digits from the MNIST Dataset (Lecun et al., 1998), for example, into a collection of points between $(0,0)$ and $(1,1)$ and run inference. We may also be able to run inference on data generated by the two other methods mentioned here. We suspect that the marginal log likelihood may be lower as the digits in all these cases are from a slightly different distribution.

Finally, given the ubiquity of point set representations in 3 and more dimensions, similar methods may find use in classifying more complex geometry. Future work may apply similar technique to classify objects or attempt rudimentary 6 DoF pose estimation.

### Conclusion

We present a simple demonstration of inverse graphics by developing a method of inferring and reconstructing digits from noisy 2D point set images. Seeking an intuitive methodology, we using importance sampling with data generation using 2D Gaussian priors. We demonstrate success on the limited case of data rendred with our data generating process, but provide extensions to generalize to more expressive data. We also provide two previous attempts to model digits and insights into why these failed and how more successful priors may be designed. We hope this work is an interesting adventure into the world of probabilistic inverse graphics and may provide intuition on how more complex probabilistic inverse graphics systems can be designed.

## References

Cusumano-Towner, M. F., Saad, F. A., Lew, A. K., & Mansinghka, V. K. (2019). Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation* (pp. 221–236). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/3314221.3314642` doi: 10.1145/3314221.3314642

Gothoskar, N., Cusumano-Towner, M., Zinberg, B., Ghavamizadeh, M., Pollok, F., Garrett, A., … Mansinghka, V. K. (2021). 3dp3: 3d scene perception via probabilistic programming.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition.*

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023, July). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, *42*(4).

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. Retrieved from `http://arxiv.org/abs/1706.03762`

Zhou, G., Gothoskar, N., Wang, L., Tenenbaum, J. B., Gutfreund, D., Lázaro-Gredilla, M., … Mansinghka, V. K. (2023). 3d neural embedding likelihood: Probabilistic inverse graphics for robust 6d pose estimation.