Singapore International Mathematical and Computational Challenge 2024 Endeavor Challenge

May 20, 2024

Contents

Preamble: Single Particle Imaging (SPI)

The predominant way to resolve the structure of fragile single proteins is by scattering probes (e.g., x-ray photons, or electrons). These probes are extremely energetic, thus it is physically impossible to probe such proteins without damaging these proteins. Because of this damage, it is hence impossible to repeatedly interrogate **the same single protein** presented at various orientations, as you would say with a patient in a medical CT scan, to obtain its 3D structure.

How then can we obtain the 3D structure of proteins? Experimentally, a prerequisite is to arrest the proteins' motion at the atomic length scale during exposure to the imaging probe. One way to arrest particle motion is to "freeze the motion" of these proteins in time! The idea is to illuminate each protein with a separate pulse of ultrafast (i.e., femtosecond, $\sim 10^{-15}$ s, pulses) x-ray laser. Consider how it takes $\sim 10^{-17}$ s for **a single photon** to traverse a nanometer sized protein. During this short time, the protein's atoms (at room temperature) are essentially static.

How then do we obtain the 3D structure of the protein? Through single particle imaging (SPI)! The idea is to illuminate many proteins singly: where we collect limited 2D structural information from one protein at a time. When we collect enough such information from many copies of the protein at different orientations, we should be able to infer the average protein's 3D structure.

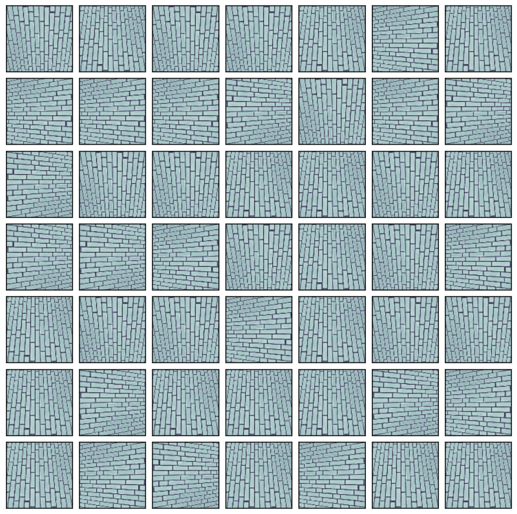
There is, however, a problem: we **do not know** the orientation of each "copy" of the protein! Each protein's orientation is hidden from us and hence known as latent parameters. Without knowing the orientation of each protein, we cannot put together the 3D structure of the average protein. In a sense, it is like trying to piece together a large jigsaw puzzle except in 3D orientation space.

Of course, inferring a single protein's hidden orientation from a single diffraction measurement is impossible. However, we might be able to infer the relative orientation between two proteins given only their respective diffraction measurements. More generally, we can use machine learning to infer the relative orientations among different proteins.

A low-dimensional toy problem.

Scientists often create simplified **toy** problems that embody the essence of the challenge in the actual research problem they are working on. These toy problems help us focus on solving the thorniest parts of the challenge without being encumbered by all of its associated realities. Coming up with these toy problems can be a bit of an art (and fun as well!).

In this toy problem, you will be given up to 100,000 2D patterns. Each pattern comprises $L \times L$ pixels. Each 2D pattern comes from the same "master image", except for a crucial detail: each pattern is a randomly rotated version of the master image.


(a) Patterns not colored by orientations.

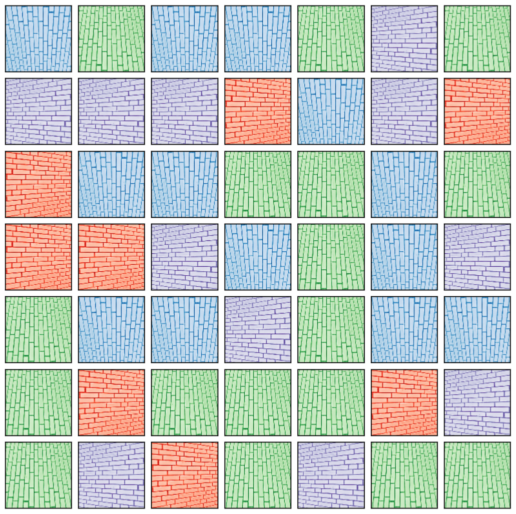

(b) Patterns colored by their orientations.

Figure 1: Example of a toy problem of SPI. 1a: You are given forty-nine patterns drawn from the same master image, except each at a random, hidden rotation (0, 90, 180, 270 degrees). 1b: You will need to uncover these patterns' orientations.

There are only four possible *clockwise* rotations (0, 90, 180, 270 degrees). However, just like the SPI problem, you do not know the rotation of each pattern.

Determining the orientation of each pattern is a very challenging problem. Consider the case of just 25 patterns, each with four possible orientations. The total number of combinations of orientations is already $4^{25} \approx 10^{15}$. Exhaustively checking which of these trillion combinations is 'best' is impractical even with the help of a computer. Yet you and I can easily solve this with visual inspection (see Task 1.).

In this challenge, you will explore algorithms that can help you classify these patterns into its four orientation classes without exhaustive searching. These algorithms will allow you to scale to very large number of patterns. For example, in Task 4. you consider up to the case of 1,000 *noisy patterns* (with $\approx 10^{6,000}$ combinations)[1], which are impossible to solve with visual inspection.

[1]The ultimate Task 7. goes up to 100,000 patterns, with $\approx 10^{60,000}$ combinations!

Reading the data.

The matrices for this challenge are stored in a Python file that you can read using the following code.

```
loaded = np.load('endeavor.npz')
task1 = loaded["task1"]
task2 = loaded["task2"]
task3 = loaded["task3"]
task4 = loaded["task4"]
task6a = loaded["task6a"]
task6b = loaded["task6b"]
task7a = loaded["task7a"]
task7b = loaded["task7b"]
```

The shapes of these matrices for the various tasks are as follows:

```
for k in loaded.keys():
    print(f"key {k} has shape: {loaded[k].shape}")

>>>>key task1 has shape: (25, 18, 18)
>>>>key task2 has shape: (100, 1296)
>>>>key task3 has shape: (1000, 1089)
>>>>key task4 has shape: (1000, 2500)
>>>>key task6a has shape: (1556153,)
>>>>key task6b has shape: (1556153,)
>>>>key task7a has shape: (2737941,)
>>>>key task7b has shape: (2737941,)
```

A CSV version of this same data is also provided.

1. Classifying a few noiseless patterns [10 points].

To build intuition, let us start simply with 25 idealized patterns in `numpy` array `task1`, like the ones shown in Figure 2. This array has shape (25 patterns, 18x18 pixels in each pattern).

Can you classify each pattern according to its relative orientations with respect to the first pattern (blue pattern in the image below)?

(a) **[4 points]** Plot the reference pattern (first pattern in `task1`) in its four clockwise rotations: 0, 90, 180, 270 degrees. *I strongly suggest you write a function that outputs four clockwise rotations of an input pattern. You will find this function useful in the following tasks.*

(b) **[4 points]** List the number of patterns in each of the four orientations with respect to the first pattern (clockwise rotation 0, 90, 180, 270 degrees).

(c) **[2 points]** Describe your approach in 300 words or fewer.

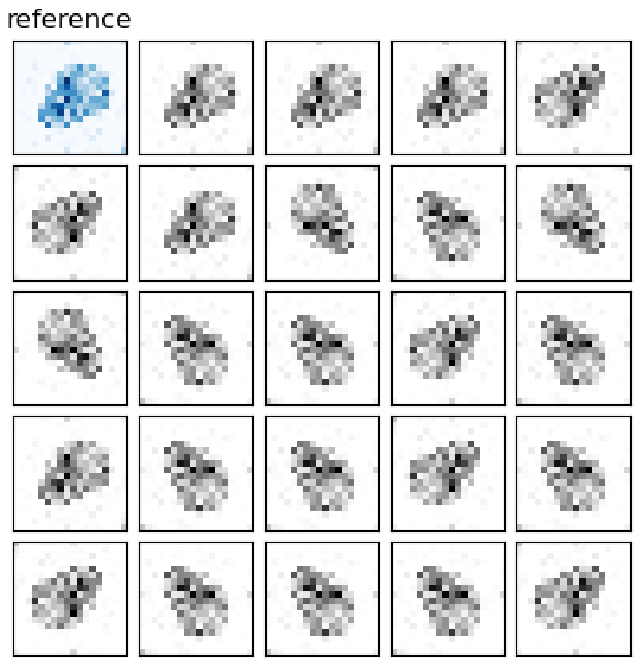


Figure 2: Twenty-five patterns in `task1`. Can you find the number of patterns that are rotated 0, 90, 180, and 270 degrees with respect to the first pattern (blue reference)?

2. Flattening 2D patterns into 1D representation [10 points].

In this toy problem, the intensity of any pixel of the master image is mutually independent of that of any other pixel. Hence, you should still be able to determine the rotation of any single pattern (formed by rotating the master image) even if the pixels of the pattern were rearranged (the same way for every pattern). In other words, you should still be able to determine if two 2D patterns share the same rotation (or not) even if they were presented as flattened 1D vectors (flattening procedure shown as green dotted line in Figure 9).

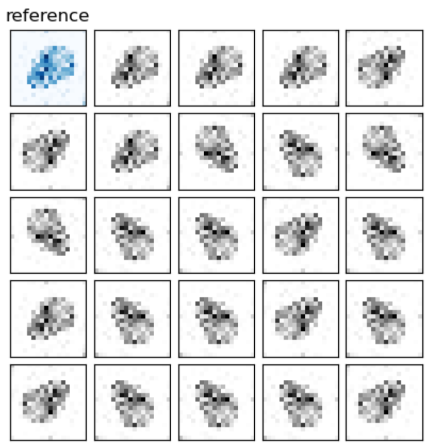
By representing individual flattened patterns as 1D vectors, we can very efficiently visualize all the patterns in a 2D array, where the rows of the array represent different patterns, and the columns of the array represent the rearranged pixel index of each pattern. **This array is called the design matrix**. See the example in Figure 3.

Classify all the 100 patterns in numpy array `task2` into four distinct orientation classes.

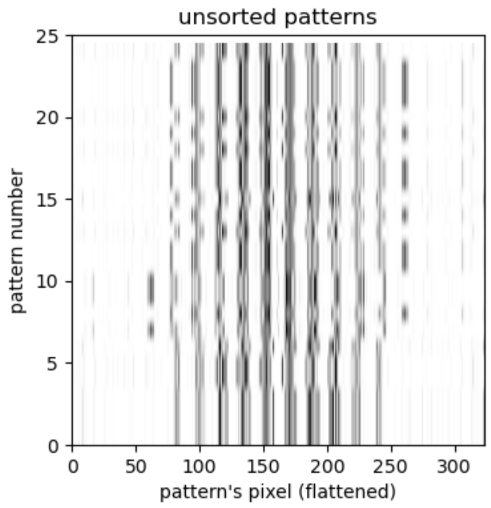
(a) **[2 points]** Render the 2D master image (e.g., using `imshow`). *Hint: each 2D pattern originally had 36x36 pixels, but was flattened to 1296 pixels.*

(b) **[4 points]** List the number of patterns in `task2` array in each of the four orientations with respect to the first pattern (clockwise rotation 0, 90, 180, 270 degrees).

(c) **[4 points]** Describe your approach in 300 words or fewer.



(a) Patterns in `task1`, which is also shown in Figure 2.



(b) Patterns in `task1` plotted as a design matrix with `matplotlib`'s `imshow` command.

Figure 3: Transforming the 2D patterns in 3a to their 1D representation as a design matrix in 3b. Each row of the design matrix represents a particular pattern, and each column represents a particular pixel's value (after rotation). Because not all the patterns are in the same orientation, the columns of the design matrix do not have the same value.

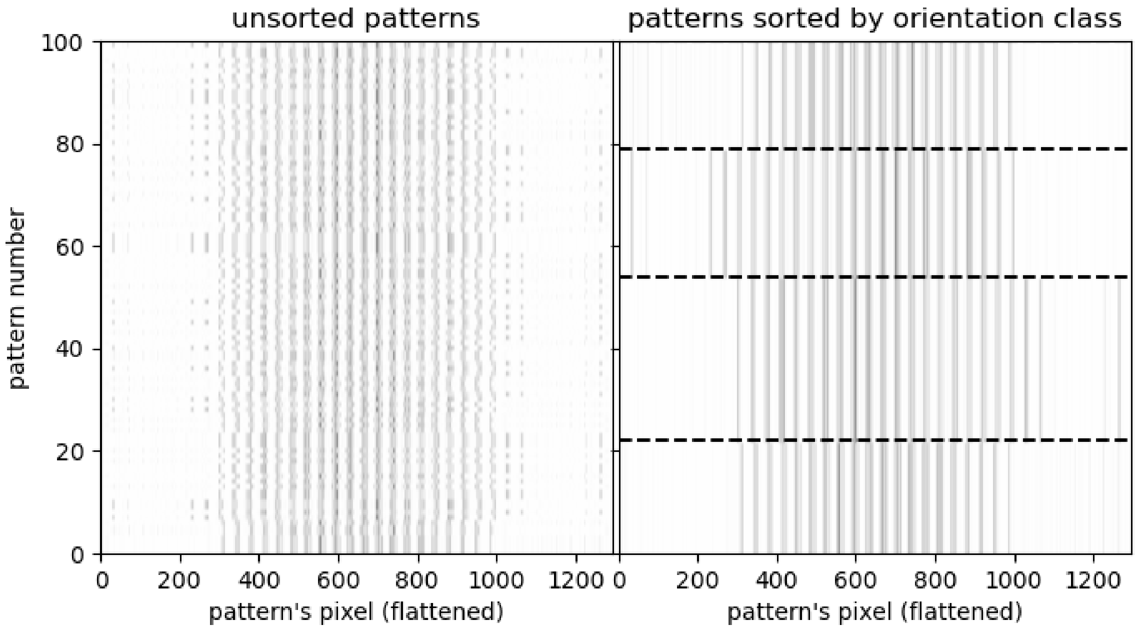


Figure 4: **LEFT**: Design matrix `task2`. **RIGHT**: By grouping the patterns (i.e., rows of design matrix) of similar orientations together (demarcated by black, horizontal, dashed lines), columns of the design matrix within the same 'orientation block' will have the same pixel value.

3. Scaling up to thousands of patterns [10 points].

In SPI, we typically collect thousands to millions of patterns. So, whatever classification strategy you used in Task 2 must now be scaled up to automatically handle many more such patterns.

Classify all the 1000 patterns in the numpy array `task3` design matrix into four distinct orientation classes.

If you get stuck here, you might want to consult the `KMeans` clustering example in the sample challenge.

(a) **[2 points]** Render the 2D master image used to create these patterns (e.g., using `imshow`). You will have to guess the shape of this 2D image to obtain the correct master image of a mystery animal.

(b) **[4 points]** List the number of patterns in each of the four orientations with respect to the first pattern (clockwise rotation 0, 90, 180, 270 degrees).

(c) **[2 points]** Visually demonstrate that you have correctly classified the patterns by plotting both your unsorted and sorted design matrix like that of Figure 5. **Important: to get full credit, you must plot this with the `BuPu` colormap.**

(d) **[2 points]** Describe your approach in 300 words or fewer.

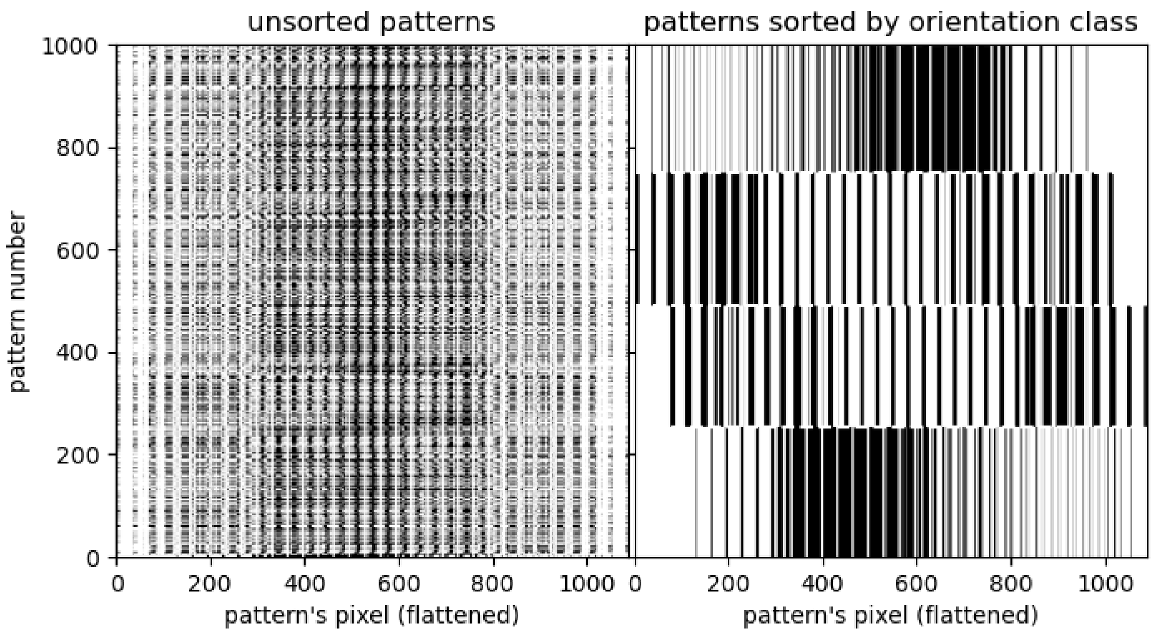


Figure 5: **LEFT**: Unsorted version of the design matrix for `task3`: each row is a pattern, and each column represents a pixel in the pattern. **RIGHT**: once you have correctly classified your patterns into the four orientation classes, you can re-sort rows of your design matrix by their orientations, giving the 'row-blocks' seen here.

4. Noisy patterns. [15 points].

Congratulations on getting this far – your algorithm can scale up to thousands of patterns! Now, things get a little more serious/realistic.

Proteins only interact very weakly with our probes (e.g., electrons and photons). This weak interaction means that each protein will only leave a weak interaction pattern on our detectors. As a result, most of the pixels on each pattern will just record zero photon/electron counts (i.e., no detectable interaction). So given such weak interactions, your earlier 1000 patterns (with good signal-to-noise ratio) from earlier tasks will look more like the Figure below.

Even though each pattern is sparse, the sum (or average) of many sparse patterns will still contain enough signal for us to discern the features (i.e., structure) of the master image. Unfortunately, without knowing the orientations of individual patterns, averaging them does not give the right 'structure' (see top left panel of Figure below).

The challenge in orientating SPI patterns is now apparent: are two patterns different because of noise or orientation? In the earlier tasks, even though the orientations of each pattern were unknown, there was enough signal in each pattern for us to decide (even unassisted by a computer) if any two patterns had the same orientation. However, with very weak (i.e., noisy) patterns, it is no longer straightforward to say if two patterns appear different because they are oriented differently or because of noise or both.

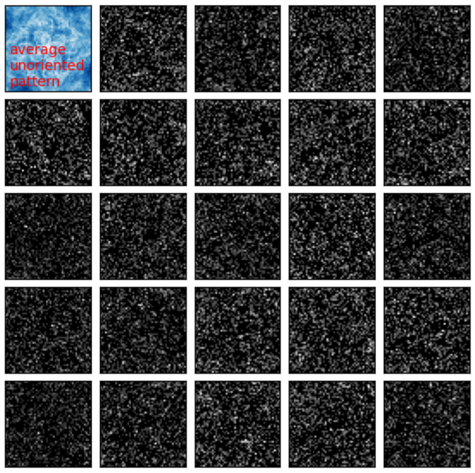


Figure 6: Twenty-four (of the one thousand) random, noisy, unoriented patterns from the design matrix `task4` shown in monochrome. Since the orientation of each pattern is unknown, averaging them together (without first re-aligning them) gives a featureless average (blue panel on top left). Furthermore, unlike the earlier cases, each pattern is too noisy for us to immediately tell its orientation by eye.

Can you determine the true "2D master image" from the design matrix `task4`?

(a) **[2 points]** What is the average 2D pattern's total pixel value (i.e., average row-sum in the design matrix)?

(b) **[3 points]** Render the average (50x50 pixel) 2D pattern if you assumed they were all the same orientation as the first pattern (similar to the first panel in Figure 6.).

(c) **[8 points]** Render each of the four orientation class averages as a 2D image. If you have correctly identified the orientations of most of the 1000 patterns, then each of the 2D class averages should look like a noisy, rotated photograph of another mystery animal.

- First, determine the most likely orientations of each pattern in the design matrix `task4`.
- Then, gather all the rows of the design matrix that correspond to a particular orientation. Average the rows according to orientation classes. You should end up with four different sums – one for each of the four orientations.
- Reshape each of the four flattened orientation class averages (2500-long) into a 50x50-pixel 2D pattern again.

(d) **[2 points]** Describe your approach in 300 words or fewer.

5. Likelihood to succeed with noisy patterns [15 points].

How can we decide if a problem is solvable? Minimally, we should be able to determine the orientation of each pattern when the master image is given to us. If this is not possible, then the problem will be exponentially more difficult (maybe even impossible) if the master image were unknown!

Our ability to determine a pattern's orientation given the master image keenly depends on knowing the noise model. Because of quantum mechanics, the number of received photons/electrons on each pixel of a very noisy pattern will largely be approximated by a Binomial distribution. Consider the i^{th} pixel on the master image with average value $\lambda_i \ll 1$. The likelihood (i.e., a kind of probability) that a pattern, with the same orientation as the master image, will contain $k_i = 1$ or $k_i = 0$ photons is denoted:

$$P(k_i = 1|\lambda_i) = \lambda_i \; , \tag{1}$$
$$P(k_i = 0|\lambda_i) = 1 - \lambda_i \; . \tag{2}$$

Four-pixel pattern case.

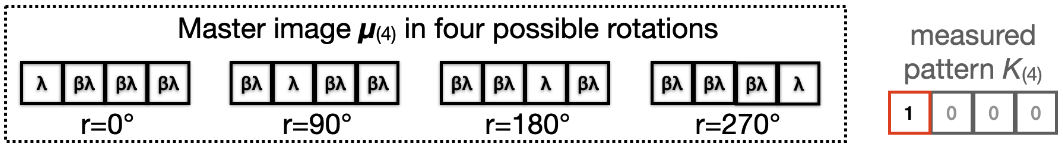

Master image $\boldsymbol{\mu}_{(4)}$ in four possible rotations
λ βλ βλ βλ
r=0°
βλ λ βλ βλ
r=90°
βλ βλ λ βλ
r=180°
βλ βλ βλ λ
r=270°
measured pattern $K_{(4)}$
1 0 0 0

Figure 7: Master image $\boldsymbol{\mu}_{(4)}$ comprises 4 pixels, which are flattened here at various orientations (inside dashed box). Each pixel has the same average value of $\beta\lambda$ with the exception of one outlier pixel whose average is λ instead. In the next exercise, we determine the likelihood that the measured pattern $K_{(4)} \equiv \{1, 0, 0, 0\}$ could arise from these four rotations.

Consider Figure 7 where the master image $\boldsymbol{\mu}_{(4)}$ contains 2x2 pixels (flattened). One of the *outlier* pixels has an average value of λ, while the other three pixels each have the same average value of $\beta\lambda$. Now, given a particular measured pattern with counts $K_{(4)} \equiv \{1, 0, 0, 0\}$, the (unnormalized) likelihood[2] that it came from the master image in orientation $r = 0°$ is,

$$\begin{aligned}\mathcal{L}(r = 0°|K_{(4)}, \boldsymbol{\mu}_{(4)}) &\equiv \Pr(k_1 = 1|\lambda)\Pr(k_2 = 0|\beta\lambda)\Pr(k_3 = 0|\beta\lambda)\Pr(k_4 = 0|\beta\lambda) \\ &= \lambda\,(1 - \beta\lambda)^3 \; .\end{aligned} \tag{3}$$

If it helps, you can think of the four pixels are independent coin flips: each pixel is a coin with a probability λ or $\beta\lambda$ of showing heads. And $k_{i=1} = 1$ is basically you observing a heads for the first coin.

(a) **[2 points]** Write a mathematical expression (in terms of variables λ and β and pure numbers) for the unnormalized likelihood $\mathcal{L}(r = 90°|K, \boldsymbol{\mu})$: that the pattern $K_{(4)} \equiv \{1, 0, 0, 0\}$ in Figure 7 arose from the master pattern in orientation $r = 90°$?

(b) **[2 points]** Write a mathematical expression (in terms of variables λ and β and pure numbers) for the ratio between the likelihoods:

$$\frac{\mathcal{L}\left(r = 0° \middle| K_{(4)}, \boldsymbol{\mu}_{(4)}\right)}{\mathcal{L}\left(r = 90° \middle| K_{(4)}, \boldsymbol{\mu}_{(4)}\right)} \; . \tag{4}$$

(c) **[2 points]** Our confidence in determining the orientation of a pattern, even if we were given the master image, is most impacted when the expression in Eqn. (4) equals one[3]. For what value of β does Eqn. (4) equal one?

Single outlier on edge pixels.

Now, consider Figure 8 where the master image $\boldsymbol{\mu}_{(16)}$ contains 4x4 pixels. One of the pixels has an average value of λ, while the remaining pixels each have the same average value of $\beta\lambda$.

[2] Strictly speaking, Equation 3 is actually a posterior distribution, which is only proportional to the likelihood when we have a uniform prior.

[3] This is true for the popular maximum-likelihood approach, which assigns each pattern to its most likely orientation. This approach becomes most ambiguous when the ratio in Eqn. (4) equals one since it is no longer clear which orientation is most likely.

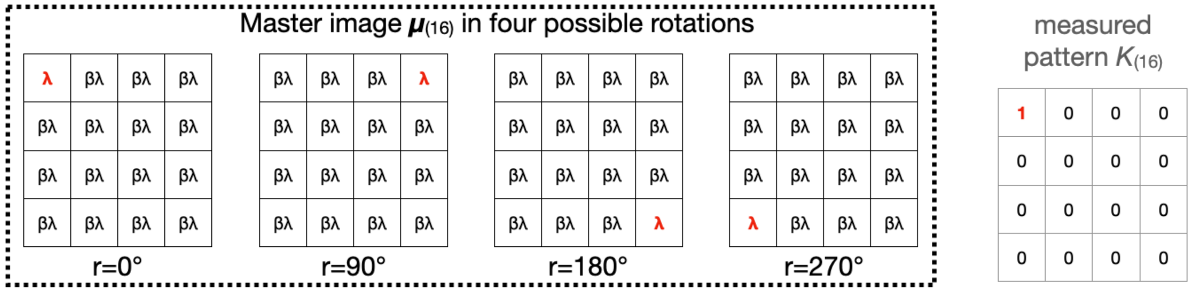

Master image $\boldsymbol{\mu}_{(16)}$ in four possible rotations
r=0°
r=90°
r=180°
r=270°
measured pattern $K_{(16)}$

Figure 8: Master image $\boldsymbol{\mu}_{(16)}$ comprises 16 pixels, shown here at various orientations (inside dashed box). Each pixel has the same average value of $\beta\lambda$ with the exception of one outlier pixel whose average is λ instead. In the next exercise, we determine the likelihood that the measured flattened pattern $K_{(16)} = \{1, 0, \ldots, 0, 0\}$ could arise from these four rotations.

(d) **[2 points]** Given the master image and pattern in Figure 8, write a mathematical expression (in terms of λ and β and pure numbers) for the ratio between the likelihoods:

$$\frac{\mathcal{L}(\text{aligned})}{\mathcal{L}(\text{misaligned})} \equiv \frac{\mathcal{L}\left(r = 0^\circ \middle| K_{(16)}, \boldsymbol{\mu}_{(16)}\right)}{\mathcal{L}\left(r = 90^\circ \middle| K_{(16)}, \boldsymbol{\mu}_{(16)}\right)} . \tag{5}$$

More outliers.

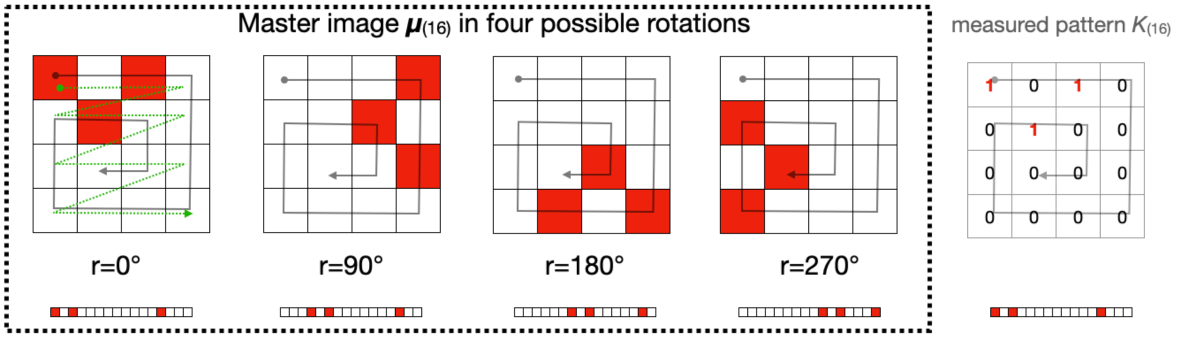

Master image $\boldsymbol{\mu}_{(16)}$ in four possible rotations
r=0°
r=90°
r=180°
r=270°
measured pattern $K_{(16)}$

Figure 9: Master image $\boldsymbol{\mu}_{(16)}$ again comprises 16 pixels, shown here at its various orientations (inside dashed box). We also show the flattened version of each rotated master image; here we flatten the pattern in a spiral sense, which is different from the usual 'left-to-right' raster (shown in green dotted line) typical of the `flatten` operation in Python. Each pixel has the same average value of $\beta\lambda$ with the exception of three outlier pixels whose average is λ instead. In the next exercise, we determine the likelihood that the measured pattern $K_{(16)} = \{1, 0, 1, \ldots, 1, 0, 0, 0\}$ could arise from these four rotations.

(e) **[2 points]** Given the master image and pattern in Figure 9, write a mathematical expression (in terms of variables λ and β and pure numbers) for the ratio between the likelihoods:

$$\frac{\mathcal{L}(\text{aligned})}{\mathcal{L}(\text{misaligned})} \equiv \frac{\mathcal{L}\left(r = 0^\circ \middle| K_{(16)}, \boldsymbol{\mu}_{(16)}\right)}{\mathcal{L}\left(r = 90^\circ \middle| K_{(16)}, \boldsymbol{\mu}_{(16)}\right)} . \tag{6}$$

Computing the log-likelihood of aligned measurements.

When the number of pixels N of the image grows large, the likelihoods that we computed earlier will start getting exponentially small. Since a computer struggles to keep such small numbers with sufficient precision, we instead calculate the log-likelihoods. Specifically for binomially distributed noise, the log-likelihood that an aligned pattern $\vec{k}$ came from a master image $\vec{\mu}$ is:

$$\ln(\mathcal{L}(\text{aligned}|\vec{\mu}, \vec{k})) = \sum_{i=1}^{N \text{ pixels}} \left(k_i \ln(\mu_i) + (1 - k_i)\ln(1 - \mu_i)\right) \ , \tag{7}$$

where the i^{th} pixel's average value is denoted μ_i, and the number of photon counts it receives is k_i. For our binomial noise distribution, we can assume that $k_i = 0$ or $k_i = 1$ without exception.

(f) **[5 points]** Given N pixels, where M of which have average value $\mu_i = \lambda$ and the remaining pixels have average value $\mu_i = \beta\lambda$, write a mathematical expression for the **average log-likelihood** in Eqn. (7) in terms of only variables N, M, β, λ and pure numbers. If your expression is correct, it will be identical to the black curves in the left panel in Figure 10.

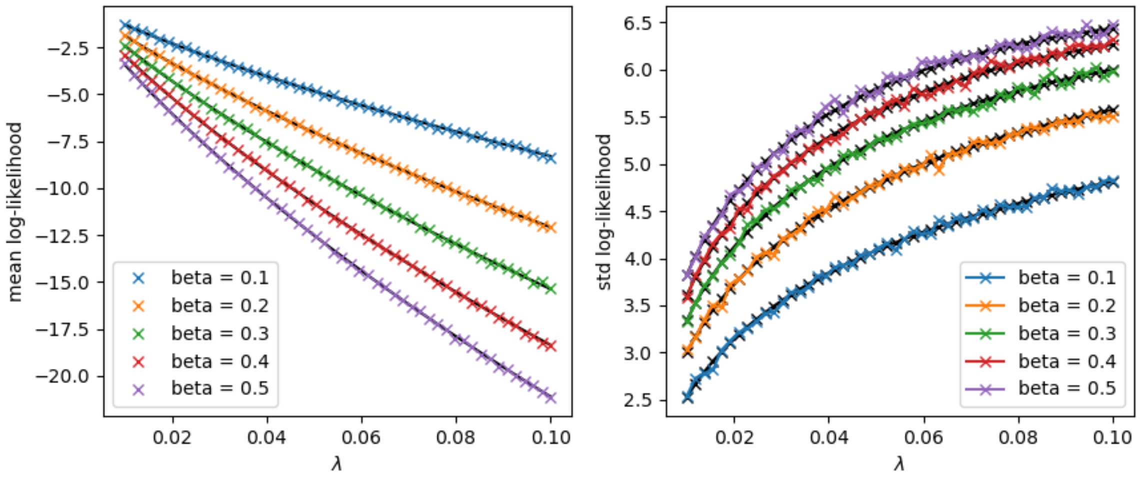


Figure 10: The average (left) and standard deviation (right) of the aligned log-likelihood in Eqn. (7) computed numerically with Monte Carlo simulations (crosses), versus the analytical expression (solid black lines) expected of you.

6. Scaling up with sparse data format [20 points].

Since each pattern gives you very weak signals, you must dramatically increase the number of measurements (i.e., patterns) to get meaningful structural information. As it stands, SPI measurements typically involve many millions of patterns, which will take up a lot of storage. Fortunately, most of these patterns do not contain useful information (i.e., have pixels with zero counts). Hence, storing these data in a sparse format is possible.

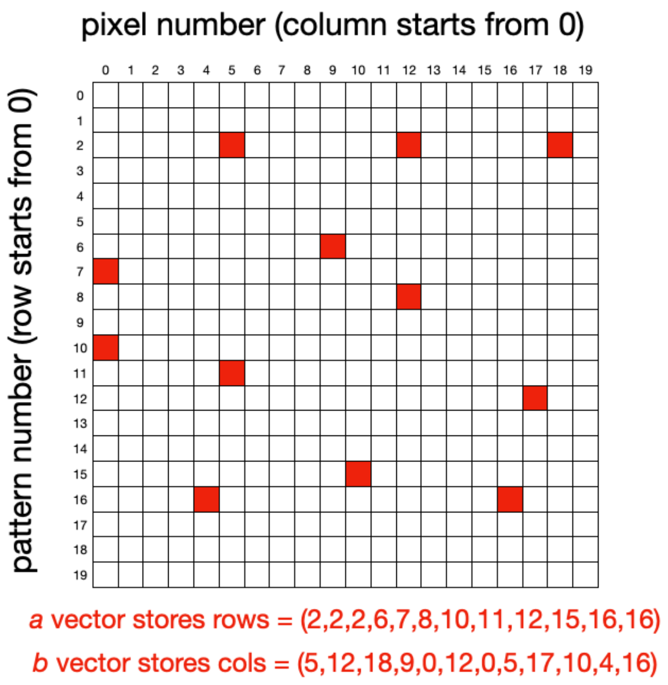

pixel number (column starts from 0)
pattern number (row starts from 0)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
a vector stores rows = (2,2,2,6,7,8,10,11,12,15,16,16)
b vector stores cols = (5,12,18,9,0,12,0,5,17,10,4,16)

Figure 11: A sparse design matrix where the red pixels are ones and the white pixels are zeros. Instead of storing the entire matrix with ones and zeros, we only store the row (vector a) and column (vector b) indices of the non-zero matrix elements.

Remember in Python, the rows and columns are indexed starting from 0, not 1.

Can you determine the true "2D master image" from the design matrix `task6`? There are 65,535 patterns, each being a flattened 25x25 pixel image.

(a) **[4 points]** What is the average pattern's total pixel value (i.e., average row-sum in the design matrix)?
(b) **[4 points]** Render the average (25x25 pixel) 2D pattern if you assumed they were all the same orientation as the first pattern (similar to the first panel in Figure 6.).
(c) **[12 points]** Render each of the four orientation class averages as a 2D image.
(d) **[5 points]** Describe your approach in 300 words or fewer.

7. Sharing data amongst orientations [20 points].

Can you determine the true "2D master image" from the design matrix `task7`? There are 100,000 patterns, each being a flattened 25x25 pixel image.

(a) **[15 points]** Render your reconstruction of the 2D master image. It should resemble a (somewhat) famous person.

(b) **[5 points]** Describe your approach in 500 words or fewer.

Hint: The number of given patterns is a critical aspect of this problem. As it stands, each orientation class will be too noisy if it is composed from only a quarter of the total number of patterns. This problem actually can be solved, but you will need to make sure that every orientation class is informed by all the patterns (rather than just a quarter of them).