A Computational Model of Synthetic Vision: Reconstructing Neuralink's Blindsight Architecture in a Spiking Visual Cortex

Alexander Speer
aes2376@columbia.edu

January 14, 2026

1 Introduction

Modern computer vision, as we know it, is largely dominated by the ideas underlying deep neural networks, yet biological visual systems seem to be able to solve similar tasks while employing very different architectures and representations. It is the goal of this project to investigate just how far one can push a biologically grounded model of the primary visual cortex (V1) toward real-time performance on a natural video, while preserving some key neuroscientific principles like the orientation columns, layered cortical structure of V1, and spike-based communication.

The system developed in this work is a computational model of the V1 system that operates based on a live video stream. More specifically, a Raspberry Pi camera is going to be sending frames at 320x240 resolution and 15 frames per second to a host machine (M1 Macbook Air), where a multi-stage pipeline will be transforming pixels into spikes, propagating them through a laminar V1 model that uses thousands of leaky integrate-and-fire (LIF) neurons, and decoding the resulting activity into orientation maps that highlight the edges and local orientation structure of the scene.[1]

The model created for this task is based on a published V1 orientation column model (referred to as MDPI2021) but reimplemented with substantial architectural and numerical modifications to enable the model to achieve near real-time performance while remaining faithful to the core biological organization of V1: four orientation columns (0°, 45°, 90°, 135°); four cortical layers (L4, L2/3, L5, L6) per column; and thousands of spiking neurons all connected in a feedforward hierarchy.

1.1 Motivation

The motivation for this project is closely aligned with the idea of *blindsight*-style neural prosthetics– devices able to bypass damaged parts of the retina or optic nerves and directly stimulate the visual cortex with synthetic patterns of activity that encode information about our visual world. In order to make such devices trustworthy and interpretable, one must generate stimulation patterns that are constrained by known cortical microcircuitry rather than arbitrary deep networks.

Additionally, from a deep learning and computer vision perspective, this project is aims to bridge between some of the standard convolutional feature extraction techniques and biological spiking neural computation. Gabor filters, latency encoding, and LIF neurons are all going to provide a biologically-inspired and interpretable alternative to the theories mentioned in class, such as the learned convolutional kernels and the ReLU activations.

[1] Our implementation is pure Python that uses NumPy and OpenCV and does not rely on any compiled neuroscience simulators.

1.2 Problem Statement

The central question guiding this project is:

> Is it possible to build a real-time, spike-based synthetic V1 that can transform live camera input into orientation-selective cortical responses, while maintaining a biologically grounded architecture and achieving a frame rate of around 5–7 frames per second?

Concretely, the system must:

- Receive the live video from the Raspberry Pi + 12MP camera.
- Extract orientation-selective features using the Gabor filters arranged in a retinotopic grid.
- Encode those features as spike trains using the concept of latency coding.
- Propagate spikes through a multi-layer orientation column model using LIF neurons.
- Take the output of those LIF neurons and decode the resulting activity into both orientation maps and strength maps that can qualitatively match the input edge structure of the image contained in the raw Pi video.
- Run fast enough to support near real-time interaction ($\approx$ 5–7 frames per second)

1.3 Contributions

The main contributions of this project are:

1. **Real-time synthetic V1:** A complete pipeline from the camera to the cortical orientation maps that is approximately 5–7 frames per second on a Macbook Air, compared to the roughly 0.02 frames per second we were getting when attempting to fully recreate the reference MDPI2021 model implementation (a 330x speedup).
2. **Biologically grounded architecture:** Creating a four-column, four-layer V1 model with 3,228 LIF neurons (807 neurons per column) and a throughline of feedforward connectivity that attempts to mirror the canonical LGN $\rightarrow$ L4 $\rightarrow$ L2/3 $\rightarrow$ L5 $\rightarrow$ L6 pathway.
3. **Analysis of design tradeoffs:** A discussion at the end of the paper that mentions how simplifying neuron models ends up affecting the resulting biological realism of the model.

1.4 Document Structure

The remainder of this report is going to be structured as follows: Chapter 2 reviews the relevant background in the biological vision and spiking neural computation, including Gabor filters, latency coding, and LIF neuron dynamics. Chapter 3 presents the full pipeline, mathematical formulations, and implementation details, reporting on the performance, firing rates, and nature of the orientation map quality. Finally, the concluding chapter summarizes the main findings, limitations of our setup, and directions that future work might take.

2 Background and Related Work

This chapter seeks to introduce the neuroscientific and computational concepts that underlie the system. These parts can be categorized as the architecture of the primary visual cortex (V1), Gabor filters as a model of simple cell receptive fields, the spike-based neural codes with latency encoding, and the LIF neuron model used in the implementation. This section is also going to briefly situate this work relative to the MDPI2021 reference model mentioned previously and other standard deep learning approaches.

2.1 Biological Visual Pathway and V1 Organization

In the primate species, all of the visual information flows from the retina to the lateral geniculate nucleus (LGN) that is located in the thalamus, and then sent to the primary visual cortex (V1). V1 is going to be organized into *orientation columns* and *layers*:

- **Retinotopy**: Neighboring points in the visual field end up being mapped to neighboring neurons on the cortical surface.
- **Layers**: Layer 4 (L4) receives the majority of the input from the thalamus' LGN, Layer 2/3 (L2/3) just projects information to the higher visual areas, and Layers 5 and 6 (L5, L6) all just send outputs to some of the subcortical structures and feedback targets.
- **Orientation selectivity**: Neurons in a given column prefer edges of a particular orientation (ex. vertical or horizontal) and are organized smoothly across the cortex.

The model implemented seeks to capture this structure at a rather coarse scale, meaning each orientation column represents one of the preferred orientations (0°, 45°, 90°, 135°). Within each of the columns, the neurons are going to be arranged in retinotopic grids for each respective layer (ex. 12x12 in L4 and L2/3), and spikes propagate feedforward from L4 to L2/3 to L5 to L6.

2.2 Gabor Filters and Simple Cell Receptive Fields

Gabor filters are thought to be a standard model of V1 simple cell receptive fields. As such, A 2D Gabor kernel at orientation θ can be written as

$$G(x, y; \theta) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda}\right), \tag{1}$$

where

$$x' = x\cos\theta + y\sin\theta, \tag{2}$$

$$y' = -x\sin\theta + y\cos\theta. \tag{3}$$

From this, we find the parameters λ (wavelength), σ (Gaussian envelope width), and γ (aspect ratio) end up controlling the spatial frequency, size, and elongation of the resulting filter. We can then assume that convolving an image with Gabor kernels at multiple different orientations yields an orientation-selective representation that, according to research done beforehand, is often used as a hand-crafted feature extractor in both classical computer vision and computational neuroscience.

2.3 Spike-Based Neural Codes and Latency Encoding

In these rate-based neural models, information often ends up represented by the average firing rate of the neurons over some time window. In spike-based models, the precise timing of these individual spikes can also carry information. One simple temporal code that we will be using is *latency coding*, which means stronger stimuli are encoded by shorter spike latencies.

So, given a normalized feature strength $f \in [0, 1]$ at a given grid position and orientation, the latency encoder used in this project maps f to a spike time

$$t_{\text{spike}} = L_{\max} - f \cdot (L_{\max} - L_{\min}), \tag{4}$$

where $L_{\min}$ and $L_{\max}$ define the earliest and the latest allowable spike times (here just 0ms and 100 ms). We can see that strong features (ex. $f \approx 1$) yield spikes near $L_{\min}$, while we could expect that all of the weaker features approach $L_{\max}$. It is also worth noting that features below a fixed threshold (0.5 used in the final configuration) do not emit spikes.

Additionally, a small amount of Gaussian jitter is added to all of the spike times to mimic trial-to-trial variability and prevent any semblance of pathological synchrony. In doing so, we are hoping this yields spike trains that are more directly interpretable in terms of their feature strength.

2.4 Leaky Integrate-and-Fire Neurons

As is the case with the MDPI2021 model, each of the neurons in the V1 model is a leaky integrate-and-fire (LIF) unit that also has separate excitatory and inhibitory synaptic currents. We can find that the membrane potential $V(t)$ evolves according to the following equations.

$$\tau_m \frac{dV}{dt} = -(V - V_{\text{rest}}) + I_{\text{syn}}, \tag{5}$$

where τ_m is the membrane time constant, V_{rest} is the resting potential, and I_{syn} is the net synaptic input (all the excitatory minus all the inhibitory currents). Additionally, we find that the synaptic currents obey exponential decay:

$$I_{\text{syn}}(t + \Delta t) = I_{\text{syn}}(t) \exp\left(-\frac{\Delta t}{\tau_{\text{syn}}}\right), \tag{6}$$

and are then going to be incremented whenever a presynaptic spike arrives at the cell, meaning they are then scaled by the synaptic weight.

After this, we can make use of Euler's method with a time step $\Delta t = 0.5$ms. When the membrane potential crosses a certain threshold V_{th} (ex. -50 mV or -55 mV, depending on layer), the neuron emits a spike, resets to V_{reset}, and enters a refractory period during which it is unable to spike.

2.5 MDPI2021 Reference Model

The MDPI2021 model found in an open-source research paper is a NEST-based implementation of V1 orientation columns with custom C++ neuron models (ex. LIF with intrinsic excitability and adaptive exponential IF). It includes a couple of things:

- 18x18 grid of neurons per primary layer.
- Intrinsic excitability plasticity trained offline to produce stable polychronous spike patterns.

- High temporal resolution (time step $\Delta t = 0.1$ ms).
- A heavy computational footprint, leading to simulation times of roughly 50 seconds per frame under typical conditions.

For this project, we keep some of the high-level architecture but re-implement the more computationally expensive parts in pure Python with a different set of design choices:

- Reduced grid size (12x12) to reduce the overall neuron count and the computations.
- Simplified the LIF neurons to have no plasticity or adaptation.
- Kept the feedforward connections only, so all of the lateral and inhibitory connections are disabled in the final configuration for the sake of stability.
- Used a coarser time step ($\Delta t = 0.5$ ms) and shorter warm-up and stimulus windows.

These changes are essential for real-time performance while preserving the layered structure of V1.

3 Methods and Implementation

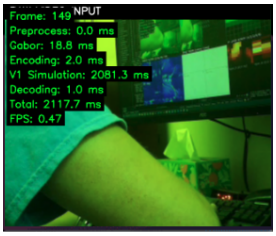
This chapter seeks to describe the complete processing pipeline from the camera frames to the orientation maps, including all of the preprocessing, Gabor feature extraction and sparsification, spike encoding, V1 simulation, and decoding. It also details some key implementation choices and presents all of the empirical results on the different firing rates, orientation maps, and performance.

3.1 System Architecture

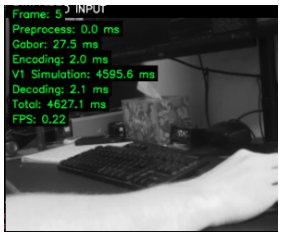
The system is organized into a five-stage pipeline:

1. **Preprocessing**: We convert all of the incoming frames to normalized grayscale images.
2. **Gabor feature extraction**: We apply four oriented Gabor filters and pool all of the responses into a 12x12 retinotopic grid per orientation.
3. **Spike encoding**: We convert the feature strengths into spike times through latency encoding.
4. **V1 simulation**: We inject spike trains into the four-column V1 model, and from there we will simulate the LIF dynamics for 150 ms (50 ms warmup, followed by a 100 ms stimulus).
5. **Decoding and visualization**: We then compute the various orientation and strength maps from Layer 2/3 firing rates and visualize them as color-coded orientation maps and activity heatmaps.

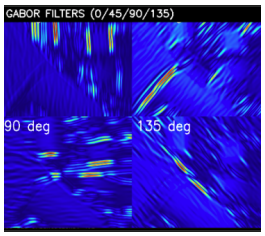
As you can see, each of the stages is implemented as a separate module and orchestrated by a central pipeline class `V1VisionPipeline`. A single entry point `realtime_pipeline.py` connects the pipeline to the Raspberry Pi video stream and manages the per-frame processing.



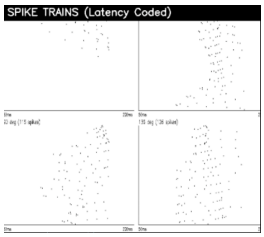
(a) Preprocessing



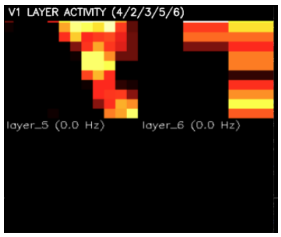
(b) Grayscale



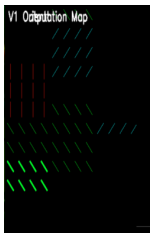
(c) Gabor filtering



(d) Spike encoding



(e) Heatmap V1 Activity Levels



(f) Final V1 Orientation Map

Figure 1: End-to-end visual processing pipeline. Top row (left to right): raw preprocessing output after normalization, grayscale conversion, and Gabor filter responses capturing oriented edge energy. Bottom row (left to right): sparsified spike events after latency encoding, Layer 2/3 V1 population activity visualized as a firing-rate heatmap, and the final decoded orientation preference map produced by the V1 model.

3.2 Preprocessing

Each frame is received as a 320×240 RGB image. Preprocessing performs the following operations:

- Conversion from RGB to a grayscale video stream.
- Gaussian blurring with a 3x3 kernel and automatically chosen σ to reduce noise.
- Contrast normalization using OpenCV's `cv2.normalize` with `NORM_MINMAX`, scaling the intensities to $[0, 255]$.

This produces a single-channel, normalized grayscale image that is robust to sensor noise and small lighting variations. (Figure 1, (a) and (b))

3.3 Gabor Feature Extraction and Sparsification

In this stage, four Gabor filters at the given orientations $\theta \in \{0°, 45°, 90°, 135°\}$ are convolved with the preprocessed image. For each of the orientations, a sliding 20x20 receptive field with 50% overlap is used to build a 12x12 feature grid.

For a given receptive field R and its orientation θ, the feature strength is defined as:

$$f_{ij}^{\theta} = \max_{(x,y) \in R_{ij}} \left| (I * G_{\theta})(x, y) \right|, \tag{7}$$

where I is the preprocessed image, G_{θ} is the Gabor kernel, and R_{ij} denotes the receptive field centered at the grid location (i, j).

Naively, we can see this produces dense feature maps in which 97–99% of cells are active above threshold. To avoid overwhelming the spike encoder and V1 model, a problem that required much troubleshooting, a sparsification pipeline is applied:

- Z-score normalization of each of the orientation maps.
- Clipping of all of the negative values.
- Percentile thresholding (ex. top 20%) so we are only keeping the strongest responses.
- Soft competition across orientations at each spatial location.

Empirically, this pipeline manages to reduce all of the active cells to roughly 10–30% per orientation and yields about 30–60 spikes per orientation per frame, which is much more manageable for the resulting downstream V1 simulation. (Figure 1(c))

3.4 Spike Encoding

For each of the grid locations and the orientation with a feature strength f_{ij}^{θ}, the encoder:

1. Normalizes f_{ij}^{θ} into $[0, 1]$.
2. Drops responses below a fixed threshold (0.5).
3. Computes the spike latency using

$$t_{ij}^{\theta} = L_{\max} - f_{ij}^{\theta} \cdot (L_{\max} - L_{\min}),$$

   with $L_{\min} = 0\,\text{ms}$ and $L_{\max} = 100\,\text{ms}$.
4. Adds a small Gaussian jitter $\mathcal{N}(0, \sigma_t^2)$ with $\sigma_t \approx 0.3\,\text{ms}$.

The result is a list of spike events that ends up taking the form

$$(\text{neuron\_id}, t_{\text{spike}}),$$

for all of the 144 neurons per orientation that pass the threshold. Ultimately, these spike trains are grouped by orientation and passed to the corresponding orientation column in the V1 model. (Figure 1(d))

3.5 V1 Model Architecture

The V1 model consists of the aforementioned orientation columns (0°, 45°, 90°, 135°), each with their own four layers. The total neuron count is 3,228 neurons, or 807 per column. Within a single column, we can find that the neuron populations are:

- **Layer 4 (Input)**: 144 spiny stellate neurons (SS) and 65 inhibitory neurons in a 12x12 grid.
- **Layer 2/3 (Primary output)**: 144 pyramidal neurons and 65 inhibitory neurons in a 12x12 grid.
- **Layer 5**: 81 pyramidal neurons and 16 inhibitory neurons in a 9x9 grid.

- **Layer 6**: 243 pyramidal neurons and 49 inhibitory neurons in a 9x27 grid.

And then the connectivity is primarily going to be feedforward:

$$\begin{aligned} \text{LGN} &\to \text{L4 SS} && (w = 5000) \\ \text{L4 SS} &\to \text{L2/3 Pyr} && (w = 120) \\ \text{L2/3 Pyr} &\to \text{L5 Pyr} && (w = 150) \\ \text{L5 Pyr} &\to \text{L6 Pyr} && (w = 150). \end{aligned}$$

All of the lateral (within-layer) and all of the recurrent (within-population) connections are disabled in the final configuration (their weights set to zero). This includes the excitatory recurrence in layers L2/3 and L5, as well as all of the inhibitory projections. Although these components do exist in the code, they are going to be turned off due to instability (in previous iterations, there was significant runaway firing or oscillations) at otherwise reasonable parameter settings.

Layer 2/3 pyramidal neurons are given slightly different parameters compared to other layers to make them more responsive:

- Lower threshold (ex. $V_{\text{th}} = -55\,\text{mV}$).
- Longer membrane time constant $\tau_m = 25\,\text{ms}$.
- A small positive bias current (ex. $20\,\text{pA}$) to ensure some level of baseline activity.

This tuning ended up being critical for preventing silent L2/3 layers. (Figure 1(e))

3.6 Simulation Loop

The V1 model is going to be simulated by using a discrete time loop with the time step $\Delta t = 0.5\,\text{ms}$. For each frame, we can see:

1. **Warmup phase** ($50\,\text{ms}$, 100 steps): The network runs without external input in order to settle into a stable state. While in continuous video mode, the warmup is only applied on the first frame that the system receives from the Raspberry Pi.
2. **Spike injection**: Latency-encoded spikes that are taken from the Gabor stage are scheduled for the appropriate L4 neurons in each orientation column.
3. **Stimulus phase** ($100\,\text{ms}$, 200 steps): At each of the time steps:
   - New spikes whose latency falls into the current time bin are going to be delivered to L4.
   - All the neuron populations then update their membrane potentials and synaptic currents.
   - Spikes are generated when thresholds are crossed, reset, and then propagated through the feedforward mechanism, along with other existing connections.

For each of the neurons, the spikes occurring during the stimulus window are counted, and the resulting firing rates in hertz are computed as

$$r = \frac{\text{spike count}}{\text{window length in seconds}}.$$

All of these rates are aggregated per layer and orientation, and finally passed to the decoder.

3.7 Decoder and Orientation Map

The decoder focuses on all of the Layer 2/3 pyramidal neurons, which are treated as the primary output of our V1 model. For each grid location (i, j) in the 12x12 retinotopic map:

- The firing rates R_{ij}^{θ} for all of the orientations $\theta \in \{0°, 45°, 90°, 135°\}$ are retrieved.
- A winner-take-all decision is then made

$$\text{Orientation}_{ij} = \arg\max_{\theta} R_{ij}^{\theta}. \tag{8}$$

- The corresponding response strength is taken as

$$\text{Strength}_{ij} = \max_{\theta} R_{ij}^{\theta}. \tag{9}$$

However, if all responses are zero for a particular location, the pixel is marked as "no response". The orientation map is then visualized by mapping each of the orientations to a distinct color (ex. red for 0°, green for 45°, blue for 90°, yellow for 135°) and the brightness of the graphic is used to indicate strength. The 12x12 map is then upsampled (ex. to 360x360) for display alongside the original frame, the Gabor feature maps, and the layer activity heatmaps. (Figure 1(f))

3.8 Implementation Details and Code Organization

The codebase ("https://github.com/alexanderspeer/dlcv-final-blindsight-project") is organized into modules that reflect all of the aforementioned stages:

- `config.py`: The central configuration for the video, grid, Gabor, spike encoding, V1 architecture, and all of the debugging statements used to improve performance.
- `gabor_extractor.py`: Gabor filter creation, convolution, and retinotopic grid construction.
- `spike_encoder.py`: The latency encoding information and the rate encoding utilities.
- `neurons.py`: Code concerning the LIF neuron and neuron population classes.
- `v1_column.py`: Single orientation column implementations with both layers and connections.
- `v1_model.py`: Multi-column V1 model, simulation loop, and result aggregation.
- `v1_decoder.py`: Orientation and strength map computation and visualization.
- `pipeline.py`: Orchestrates all of the stages for a single frame.
- `realtime_pipeline.py`: Connects pipeline to camera stream and runs the per-frame loop.

3.9 Experimental Results

3.9.1 Firing Rates and Orientation Selectivity

Since it is rather hard to measure the accuracy of the model during the real-time pipeline, using static test images with oriented edges and natural scenes may be the best way to evaluate the system. With this in mind, the system exhibits the following typical firing rates:

- **Layer 4**: 40–50 Hz average, with roughly 60% of the neurons active above 1 Hz.
- **Layer 2/3**: 10–25 Hz average, with 50% of the neurons active above 1 Hz; indicative of strong orientation selectivity across the columns.
- **Layer 5**: 5–20 Hz, moderate activity driven by L2/3.
- **Layer 6**: 2–15 Hz, stable with no runaway excitation detected.

For a representative frame, the sample mean L2/3 firing rates might look like:

$$\begin{aligned} 0^\circ &: \quad 38.7 \text{ Hz} \\ 45^\circ &: \quad 22.3 \text{ Hz} \\ 90^\circ &: \quad 15.8 \text{ Hz} \\ 135^\circ &: \quad 19.5 \text{ Hz}, \end{aligned}$$

with the dominant orientation matching the dominant edge structure we would see in its input.

3.9.2 Orientation Maps

From the decoder step, we are going to produce orientation maps in which:

- High-contrast edges lead to clear colored bands aligned with their orientation.
- Flat or uniform regions where the model yields no response.
- Complex natural scenes that produce mixed orientation patches, with some winner-take-all decisions sometimes arbitrary, where multiple of the orientations are responded to similarly.

In all of the successful cases, fewer than 50% of pixels are going to be marked as "no response", and most of the distribution of preferred orientations is reasonably balanced given the stimulus input. In some of the earlier iterations, nearly all of the pixels were "no response" due to the silent L2/3 layers (changed once weight tuning and parameter changes were implemented).

3.9.3 Performance and Frame Rate

Timing measurements per frame (excluding all of the I/O) are going to be approximately:

- Preprocessing: 1–2 ms.
- Gabor extraction: 5–10 ms.
- Spike encoding: 1–2 ms.
- V1 simulation: 100–200 ms (dominant cost).
- Decoding: 2–5 ms.
- Visualization: 20–40 ms (orientation maps, heatmaps, combined panels).

With the final optimizations consisting of time per frame being roughly 150–180 ms, corresponding to about 5–7 frames per second. The reference MDPI2021 implementation runs at approximately 0.02 frames per second (about 50 seconds per frame), showing a speedup of 330x.

Some more of the key optimizations include:

- Reducing the grid size from 18x18 to 12x12 (about 56% fewer neurons per each layer).
- Increasing the time step from 0.1 ms to 0.5 ms (5x fewer simulation steps).
- Reducing the warmup from around 400 ms per frame to 50 ms on the first frame only.
- Shortening the duration of the stimulus from 200 ms to 100 ms.
- Disabling the recurrent and inhibitory connections causing runaway excitation or oscillations.

4 Conclusion

In summary, this project attempted to implement a real-time computational model of the primary visual cortex that is able to transform live camera input into a myriad of orientation-selective cortical responses using an architecture that is biologically grounded. To accomplish this, the system combines classical computer vision techniques (such as Gabor filters) with spiking neural computation (latency encoding and LIF neurons) to produce several different interpretable orientation maps and layer-specific activity patterns.

4.1 Summary of Findings

The main findings from this project are as follows:

- A four-column, four-layer V1 model with 3,228 spiking neurons can be driven by live video and simulated fast enough to achieve approx. 5–7 frames per second on commodity hardware.
- Gabor feature extraction, followed by sparsification and latency encoding, can yield spike trains that capture the coarse orientation structure of each frame while remaining computationally tractable.
- With appropriate weight tuning and neuron parameter adjustments, Layer 2/3 can exhibit a clear orientation-selective firing pattern that can be decoded into orientation maps that are loosely aligned with the underlying edges that comprise the input.
- Achieving real-time performance requires deliberate departures from the MDPI2021 model, including simpler neuron models, reduced spatial and temporal resolution, and the deactivation of recurrent and inhibitory connections difficult to stabilize in pure Python.

From a computational viewpoint, we believe the system demonstrates an alternative pipeline in which early visual processing is able to be performed by an interpretable and biologically motivated circuit, rather than simply learned convolutional layers. This could indicate opportunities for hybrid architectures down the line, where such models would be able to provide structured input to downstream deep networks or help generate stimulation patterns for neuroprosthetic devices.

4.2 Limitations

Several limitations should be acknowledged:

- **No learning**: All weights and neuron parameters were fixed and tuned manually. There was no intrinsic excitability, learning, or synaptic plasticity as in the MDPI2021 model.

- **No inhibition or lateral connectivity**: The inhibition and recurrent connections were disabled, reducing the richness of cortical dynamics and varied cross-orientation interactions.
- **Coarse spatial and temporal resolution**: The 12x12 grid and 0.5ms time step are chosen for performance and could potentially miss finer structure present in the biological V1.
- **Single-stage pipeline**: Some of the higher visual areas (such as V2, V4, MT) and feedback pathways are notably absent, meaning the model can only compute local orientation maps rather than performing any sort of object recognition or motion analysis.

4.3 Future Work

We believe there are several promising directions for extending this work:

- **Reintroducing inhibition and recurrence**: In our opinion, with more careful parameter search, or more efficient simulation (ex. JIT compilation or C++ extensions), it could be possible to re-enable the different lateral and inhibitory connections, which would allow us to recover richer dynamics such as the cross-orientation suppression and the surround modulation.
- **Learning and plasticity**: Perhaps incorporating intrinsic excitability plasticity or some form of synaptic plasticity (ex. STDP) would allow the model to better adapt to the statistics of its input and consequently, move closer to the MDPI2021 approach.
- **Scaling and hardware acceleration**: GPU-based implementations, some form of vectorized neuron updates, or any sort of event-driven simulation could likely push the model beyond real time and enable higher resolution (ex. 18x18 or 24x24 grids).
- **Integration with deep learning**: It is not a stretch of the imagination to consider the orientation and strength maps produced by this model could be fed into a downstream CNN or transformer, providing a biologically motivated front end for traditionally complex tasks such as recognition, tracking, or segmentation.
- **Neuroprosthetics applications**: Since the entire project is modelled after a real-world Brain Computer Interface concept, perhaps with appropriate calibration and safety constraints, patterns of activity from the model could be mapped to cortical stimulation patterns for use in brain-computer interfaces targeting visual restoration.

5 Appendix

5.1 Key Configuration Parameters

Table 1 is able to summarize the main spatial and temporal parameters that were used in the final configuration.

Table 1: Grid, temporal, and spike encoding parameters.

Parameter	Value
Frame resolution	$320x240$
Grid size (L4, L2/3)	12×12
Receptive field size	20×20 pixels
Receptive field overlap	50%
Time step Δt	0.5 ms
Warmup duration	50 ms (first frame only)
Stimulus duration	100 ms
Latency range	$L_{\min} = 0\,\text{ms}, L_{\max} = 100\,\text{ms}$
Latency jitter σ_t	$\approx 0.3\,\text{ms}$
Spike threshold (features)	0.5 (normalized units)

5.2 Neuron Counts per Layer

Table 2 lists the neuron counts per layer and column.

Table 2: Neuron counts per orientation column.

Layer	Population	Count
L4	Spiny stellate (SS)	144
	Inhibitory	65
L2/3	Pyramidal	144
	Inhibitory	65
L5	Pyramidal	81
	Inhibitory	16
L6	Pyramidal	243
	Inhibitory	49
Total per column		807
Total (4 columns)		3,228

5.3 Synaptic Weights and Neuron Parameters

Table 3 summarizes all of the effective feedforward weights that were used in the final model.

Neuron parameters are largely shared across layers, with small variations in L2/3:

- Resting potential $V_{\text{rest}} = -65\,\text{mV}$.
- Reset potential $V_{\text{reset}} = -65\,\text{mV}$.
- Threshold $V_{\text{th}} = -50\,\text{mV}$ for most of the neurons, $-55\,\text{mV}$ for the L2/3 pyramidal neurons.
- Membrane time constant $\tau_m = 10\,\text{ms}$ generally, 25 ms for the L2/3 pyramidal neurons.

Table 3: Feedforward synaptic weights (final configuration).

Connection	Weight	Notes
LGN → L4 SS	5000	Strong input drive
L4 SS → L2/3 Pyr	120	Increased from 20 to avoid silence
L2/3 Pyr → L5 Pyr	150	Decreased from 800 to avoid runaway L5
L5 Pyr → L6 Pyr	150	Decreased from 1200 to avoid runaway L6
Lateral and inhibitory	0	Disabled in final configuration

- Synaptic time constants were $\tau_{\text{syn,ex}} = \tau_{\text{syn,in}} = 2\,\text{ms}$.
- Refractory period $\approx 2\,\text{ms}$.

5.4 Pseudo-code for the Full Pipeline

For the sake of completeness, the Algorithm 2 sketches the processing pipeline in pseudo-code.

Algorithm 1: Real-time V1 vision pipeline (per frame)

1. Receive RGB frame from camera.
2. Preprocess: convert to grayscale, blur, normalize.
3. For each orientation θ:
   (a) Convolve with Gabor filter G_θ.
   (b) Build 12x12 grid by max-pooling over 20x20 windows.
   (c) Apply sparsification and normalization.
4. Encode active grid cells as latency-coded spikes.
5. Inject spikes into corresponding orientation column L4.
6. Simulate V1 for warmup + stimulus duration.
7. Compute firing rates for all layers.
8. Decode Layer 2/3 rates into orientation and strength maps.
9. Visualize original frame, Gabor maps, layer heatmaps, and orientation map.

Figure 2: High-level algorithm for the real-time V1 vision pipeline.

[1]

References

[1] Alejandro Santos-Mayo et al. "A Model of the Early Visual System Based on Parallel Spike-Sequence Detection, Showing Orientation Selectivity". In: *Biology* 10.8 (2021), p. 801. DOI: 10.3390/biology10080801.