Model-Free Stochastic Perturbative Adaptation and Optimization

Gert Cauwenberghs
Johns Hopkins University
gert@jhu.edu

520.776 Learning on Silicon
http://bach.ece.jhu.edu/gert/courses/776

Model-Free Stochastic Perturbative Adaptation and Optimization

OUTLINE

- **Model-Free Learning**
  - Model Complexity
  - Compensation of Analog VLSI Mismatch
- **Stochastic Parallel Gradient Descent**
  - Algorithmic Properties
  - Mixed-Signal Architecture
  - VLSI Implementation
- **Extensions**
  - Learning of Continuous-Time Dynamics
  - Reinforcement Learning
- **Model-Free Adaptive Optics**
  - AdOpt VLSI Controller
  - Adaptive Optics "Quality" Metrics
  - Applications to Laser Communication and Imaging

The Analog Computing Paradigm

- **Local functions are efficiently implemented with minimal circuitry, exploiting the physics of the devices.**
- **Excessive global interconnects are avoided:**
  - *Currents or charges are accumulated along a single wire.*
  - *Voltage is distributed along a single wire.*

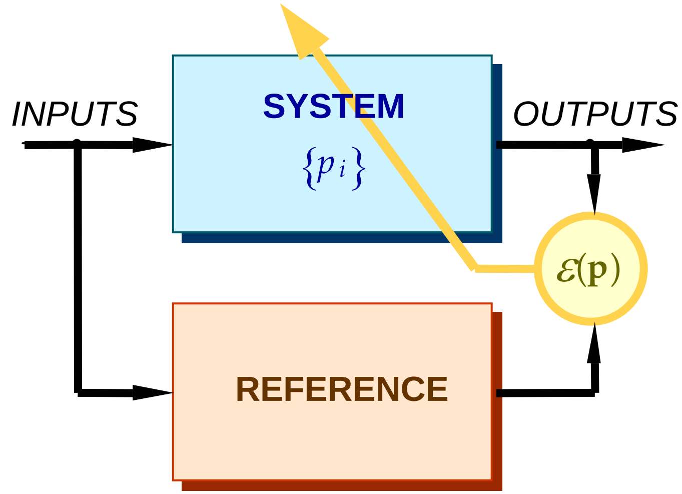
Pros:

- Massive Parallellism
- Low Power Dissipation
- Real-Time, Real-World Interface
- Continuous-Time Dynamics

Cons:

- Limited Dynamic Range
- Mismatches and Nonlinearities (WYDINWYG)

Effect of Implementation Mismatches



Associative Element:

– Mismatches can be properly compensated by adjusting the parameters p_i accordingly, provided sufficient degrees of freedom are available to do so.

Adaptive Element:

– Requires precise implementation

– The accuracy of implemented *polarity* (rather than amplitude) of parameter update increments Δp_i is the performance limiting factor.

Example: LMS Rule

A linear perceptron under supervised learning:

$$y_i^{(k)} = \sum_j p_{ij}\, x_j^{(k)}$$

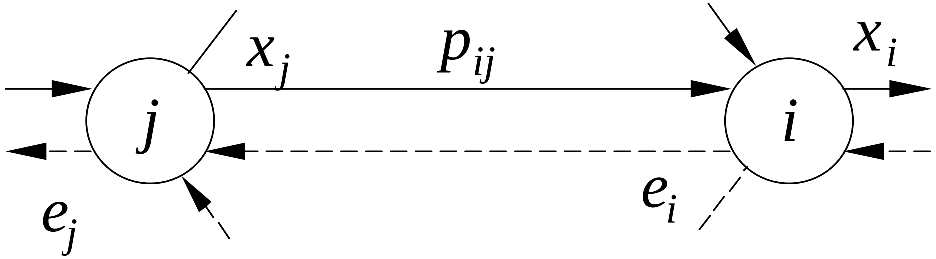
$$\varepsilon = \frac{1}{2} \sum_k \sum_j \left(y_i^{\text{target}(k)} - y_i^{(k)} \right)^2$$

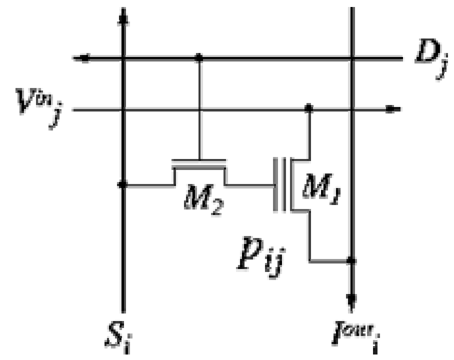
with gradient descent:

$$\Delta p_{ij}^{(k)} = -\eta \, \frac{\partial \varepsilon^{(k)}}{\partial p_{ij}} = -\eta \; x_j^{(k)} \cdot \left(y_i^{\text{target}(k)} - y_i^{(k)} \right)$$

reduces to an *incremental outer-product* update rule, with scalable, modular implementation in analog VLSI.

Incremental Outer-Product Learning in Neural Nets





Multi-Layer Perceptron: $$x_i = f(\sum_j p_{ij} x_j)$$

Outer-Product Learning Update: $$\Delta p_{ij} = \eta \; x_j \cdot e_i$$

- Hebbian *(Hebb, 1949)*: $$e_i = x_i$$
- LMS Rule *(Widrow-Hoff, 1960)*: $$e_i = f'_i \cdot \left(x_i^{\text{target}} - x_i\right)$$
- Backpropagation *(Werbos, Rumelhart, LeCun)*: $$e_j = f'_j \cdot \sum_i p_{ij} e_i$$

Gradient Descent Learning

Minimize $\mathcal{E}(\mathbf{p})$ by iterating:

$$p_i^{(k+1)} = p_i^{(k)} - \eta \frac{\partial \mathcal{E}^{(k)}}{\partial p_i}$$

from calculation of the gradient:

$$\frac{\partial \mathcal{E}}{\partial p_i} = \sum_l \sum_m \frac{\partial \mathcal{E}}{\partial y_l} \cdot \frac{\partial y_l}{\partial x_m} \cdot \frac{\partial x_m}{\partial p_i}$$

Implementation Problems:

- Requires an explicit model of the internal network dynamics.
- Sensitive to model mismatches and noise in the implemented network and learning system.
- Amount of computation typically scales strongly with the number of parameters.

Gradient-Free Approach to Error-Descent Learning

Avoid the model sensitivity of gradient descent, by observing the parameter dependence of the performance error on the network directly, rather than calculating gradient information from a pre-assumed model of the network.

Stochastic Approximation:

- Multi-dimensional Kiefer-Wolfowitz *(Kushner & Clark 1978)*
- Function Smoothing Global Optimization *(Styblinski & Tang 1990)*
- Simultaneous Perturbation Stochastic Approximation *(Spall 1992)*

Hardware-Related Variants:

- Model-Free Distributed Learning *(Dembo & Kailath 1990)*
- Noise Injection and Correlation *(Anderson & Kerns; Kirk & al. 1992-93)*
- Stochastic Error Descent *(Cauwenberghs 1993)*
- Constant Perturbation, Random Sign *(Alspector & al. 1993)*
- Summed Weight Neuron Perturbation *(Flower & Jabri 1993)*

Stochastic Error-Descent Learning

Minimize $\mathcal{E}(\mathbf{p})$ by iterating:

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \mu \hat{\mathcal{E}}^{(k)} \pi^{(k)}$$

from observation of the gradient in the direction of $\pi^{(k)}$:

$$\hat{\mathcal{E}}^{(k)} = \tfrac{1}{2}\Big(\varepsilon(\mathbf{p}^{(k)} + \pi^{(k)}) - \varepsilon(\mathbf{p}^{(k)} - \pi^{(k)})\Big)$$

with random uncorrelated binary components of the perturbation

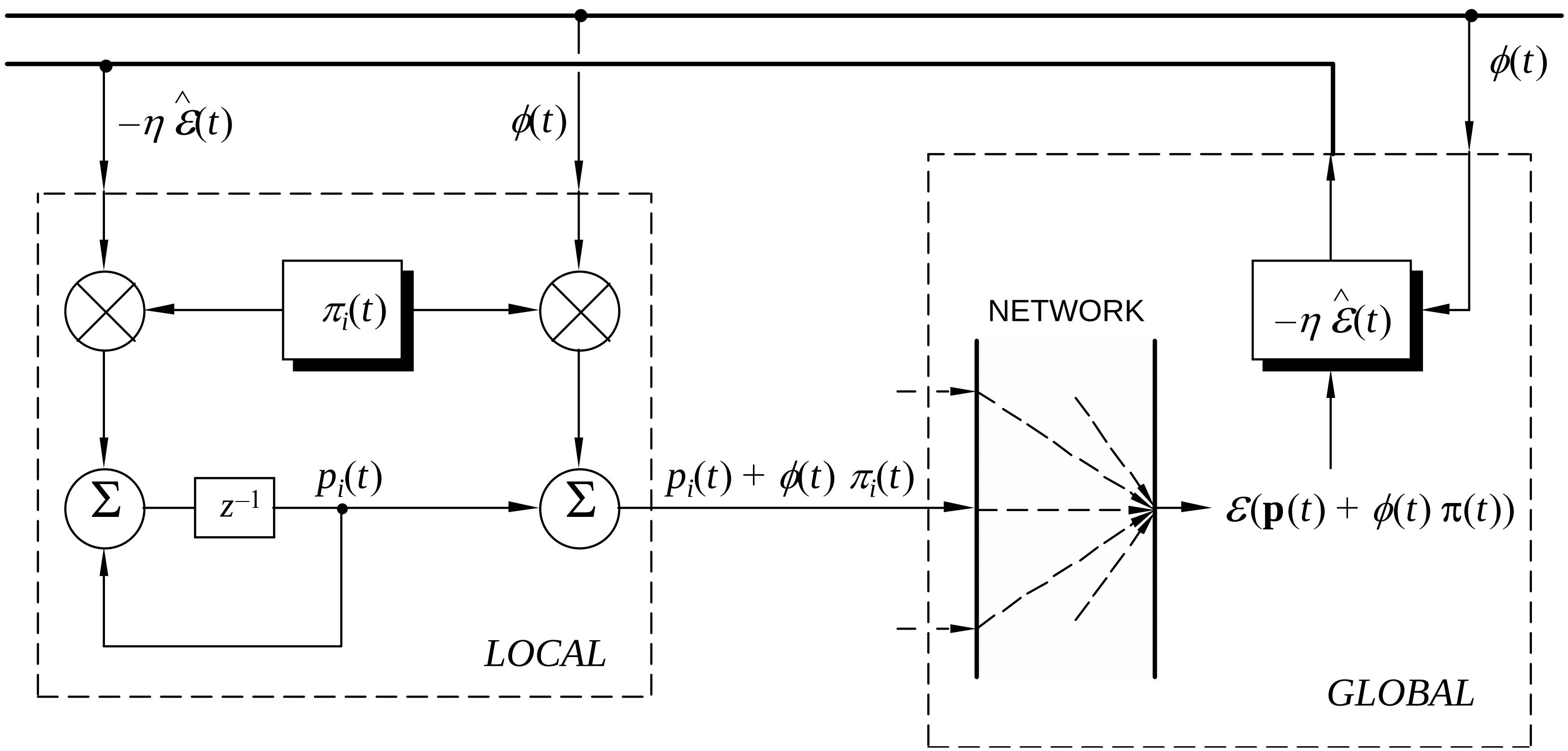
vector $\pi^{(k)}$: $\quad \pi_i^{(k)} = \pm\sigma \;;\quad E(\pi_i^{(k)} \pi_j^{(l)}) \approx \sigma^2 \,\delta_{ij} \delta_{kl}$

Advantages:

- No explicit model knowledge is required.
- Robust in the presence of noise and model mismatches.
- Computational load is significantly reduced.
- Allows simple, modular, and scalable implementation.
- Convergence properties similar to exact gradient descent.

Stochastic Perturbative Learning

Cell Architecture

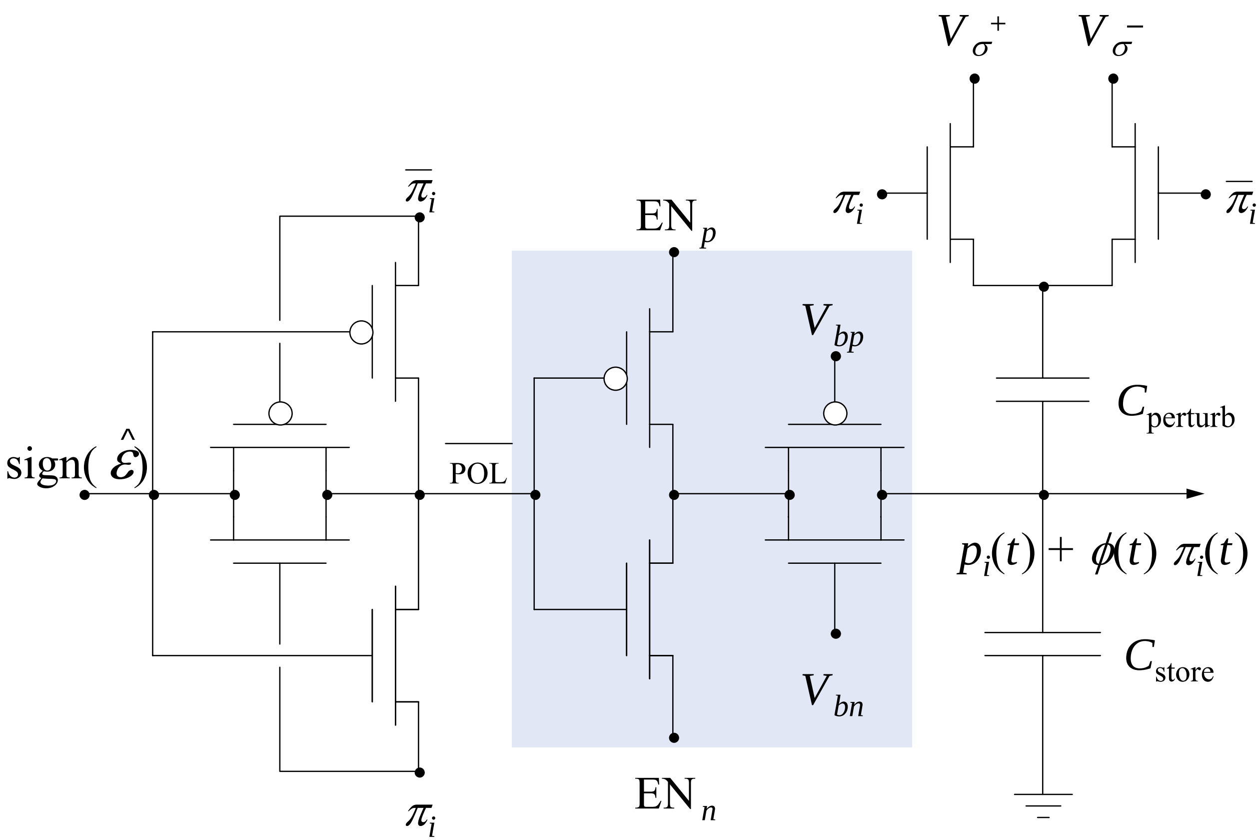


$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \mu\hat{\varepsilon}^{(k)}\boldsymbol{\pi}^{(k)}$$

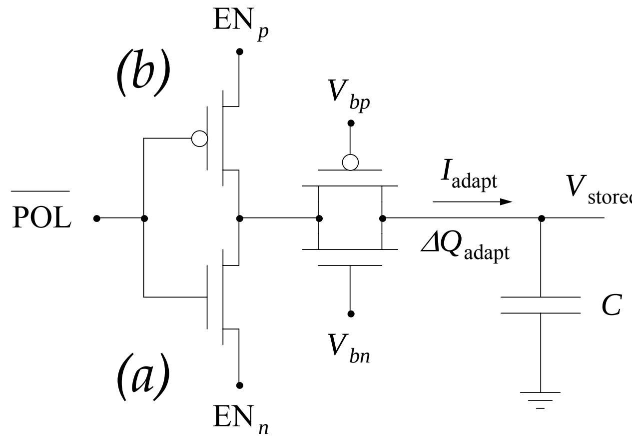
$$\hat{\varepsilon}^{(k)} = \tfrac{1}{2}\Big(\varepsilon(\mathbf{p}^{(k)}+\boldsymbol{\pi}^{(k)}) - \varepsilon(\mathbf{p}^{(k)}-\boldsymbol{\pi}^{(k)})\Big)$$

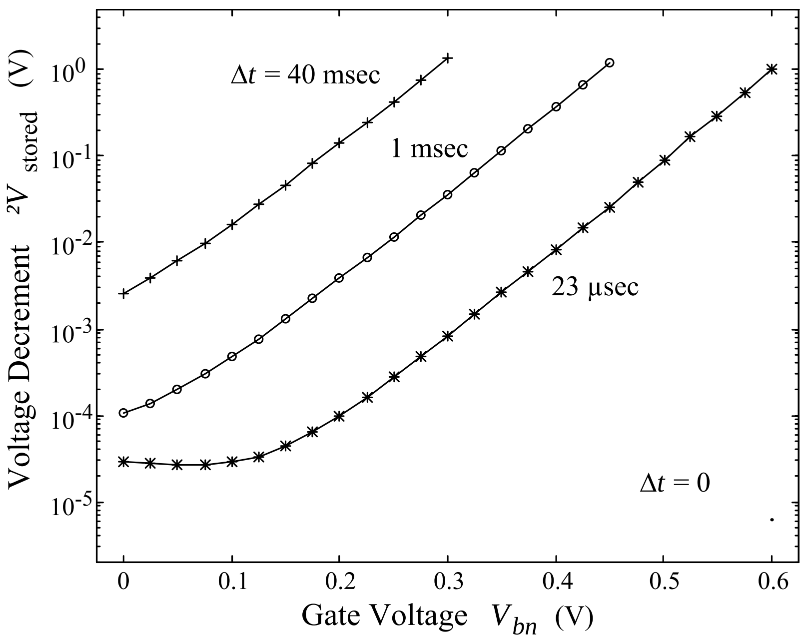
Stochastic Perturbative Learning

Circuit Cell

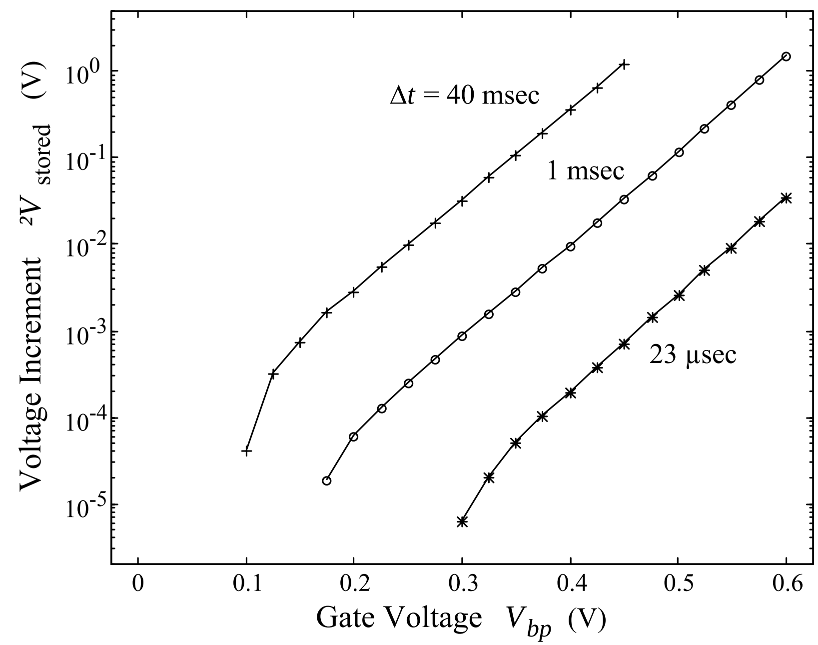


Charge Pump Characteristics



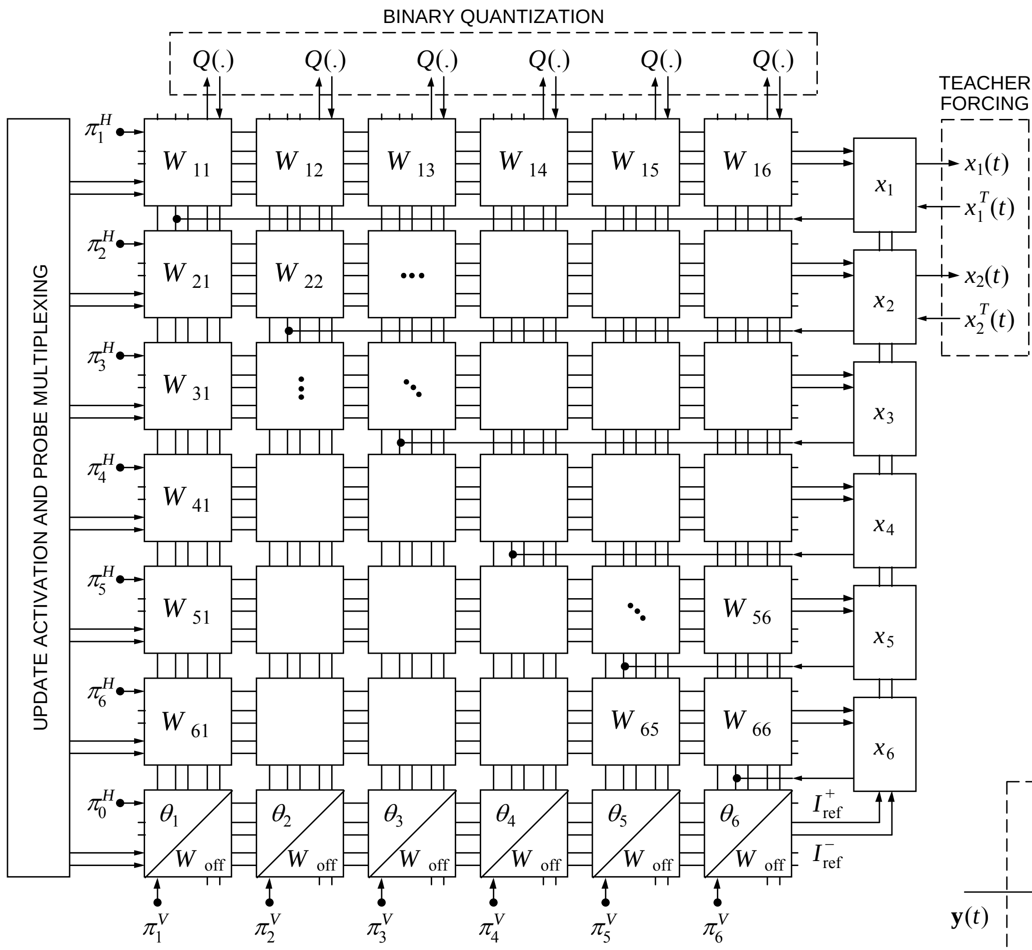


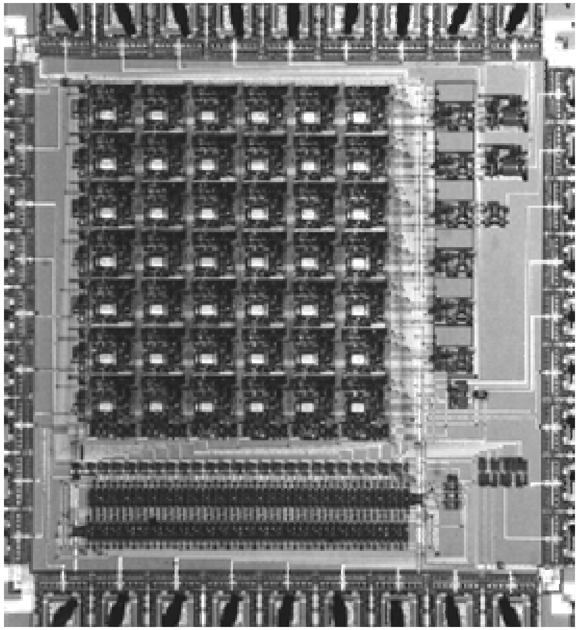
(a)

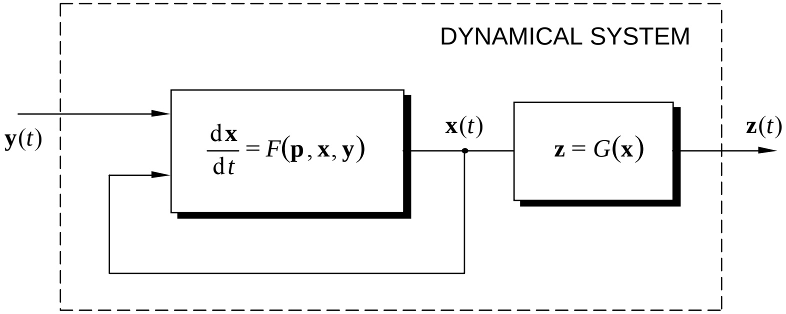


(b)

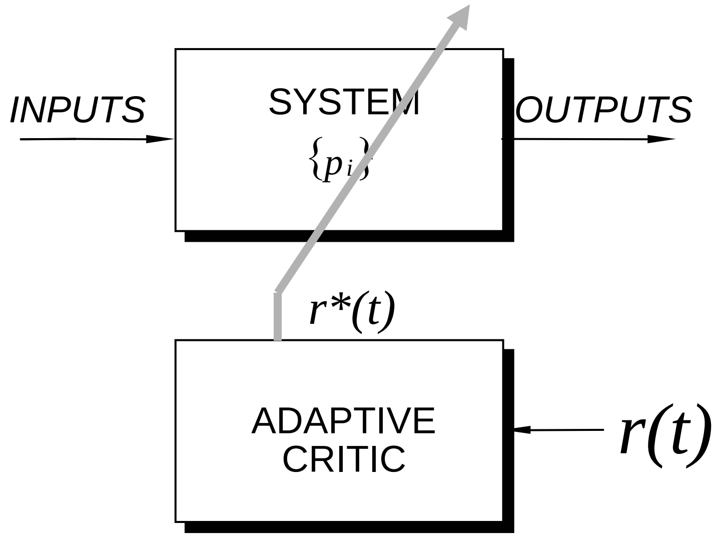
Supervised Learning of Recurrent Neural Dynamics







The Credit Assignment Problem
or How to Learn from Delayed Rewards



- External, discontinuous reinforcement signal *r(t)*.
- Adaptive Critics:
  - *Discrimination Learning (Grossberg, 1975)*
  - *Heuristic Dynamic Programming (Werbos, 1977)*
  - *Reinforcement Learning (Sutton and Barto, 1983)*
  - *TD(λ) (Sutton, 1988)*
  - *Q-Learning (Watkins, 1989)*

Reinforcement Learning
(Barto and Sutton, 1983)

Locally tuned, address encoded neurons:

$$\chi(t) \in \{0,\ldots 2^n-1\} : \text{ } n\text{–bit address encoding of state space}$$

$$y(t) = y_{\chi(t)} : \text{ classifier output}$$

$$q(t) = q_{\chi(t)} : \text{ adaptive critic}$$

Adaptation of classifier and adaptive critic:

$$y_k(t+1) = y_k(t) + \alpha\, \hat{r}(t)\, e_k(t)\, y_k(t)$$

$$q_k(t+1) = q_k(t) + \beta\, \hat{r}(t)\, e_k(t)$$

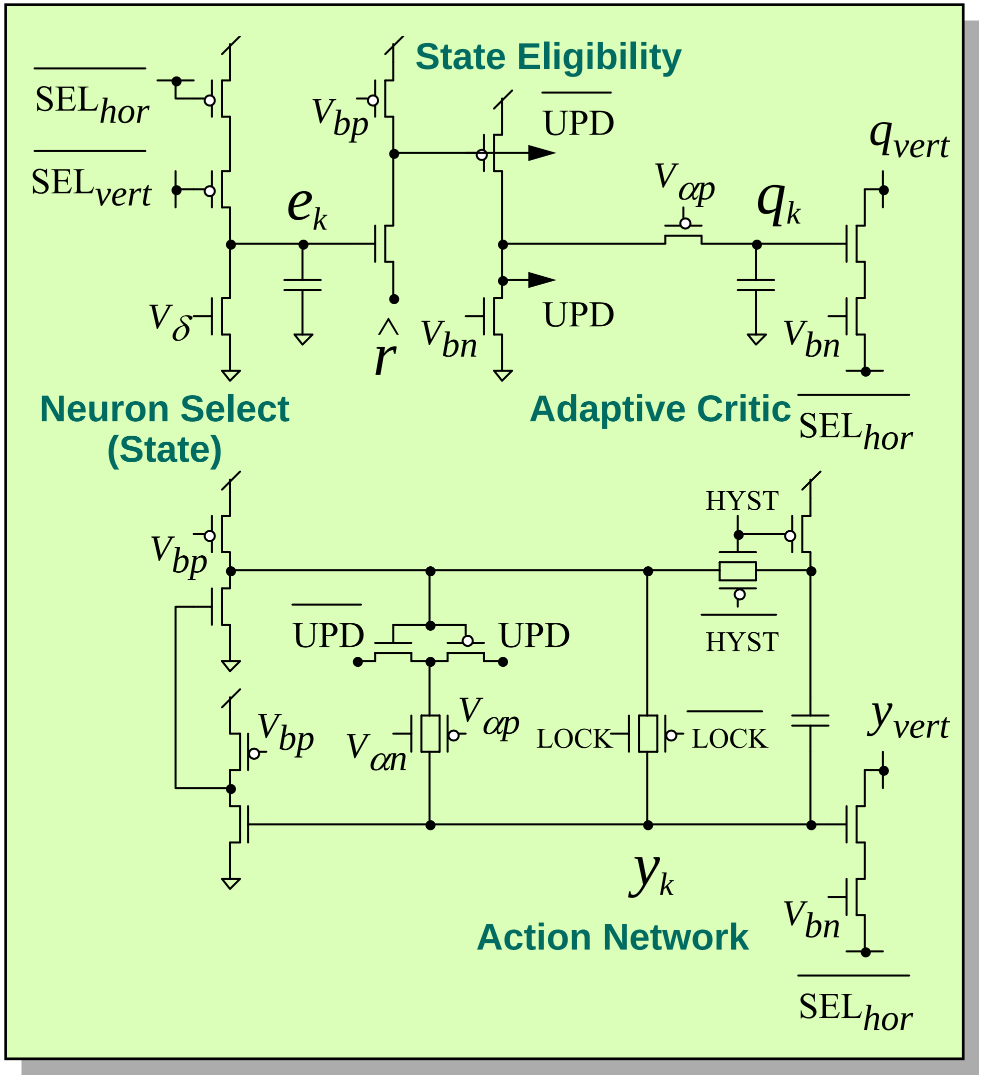
– eligibilities:

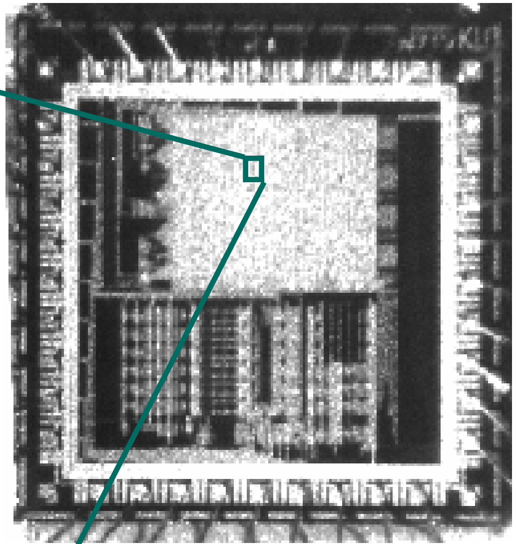
$$e_k(t+1) = \lambda\, e_k(t) + (1-\lambda)\, \delta_{k\, \chi(t)}$$

– internal reinforcement:

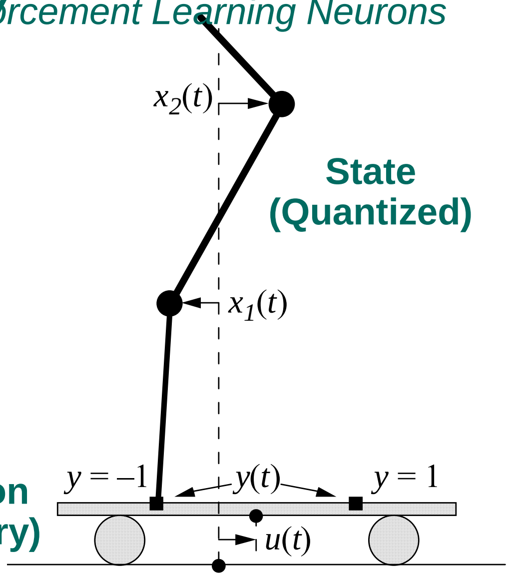
$$\hat{r}(t) = r(t) + \gamma\, q(t) - q(t-1)$$

Reinforcement Learning Classifier for Binary Control

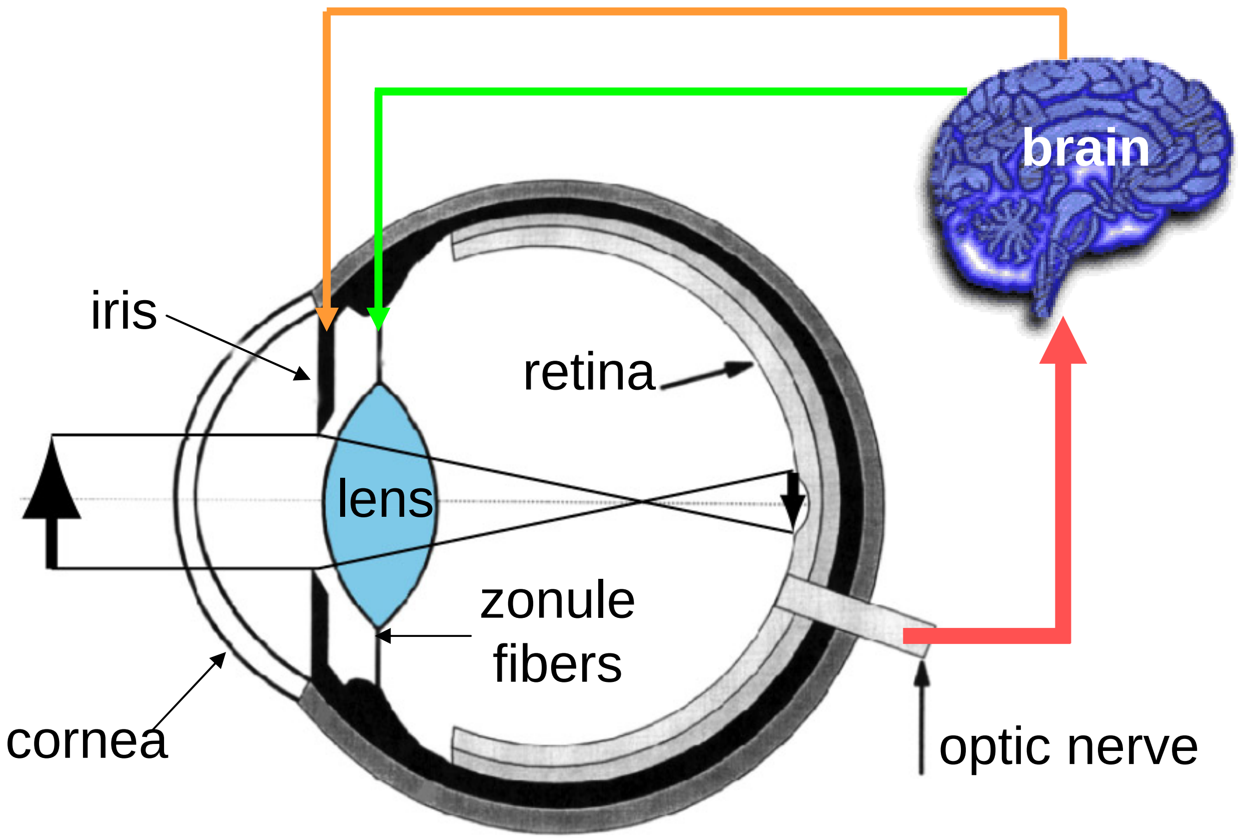




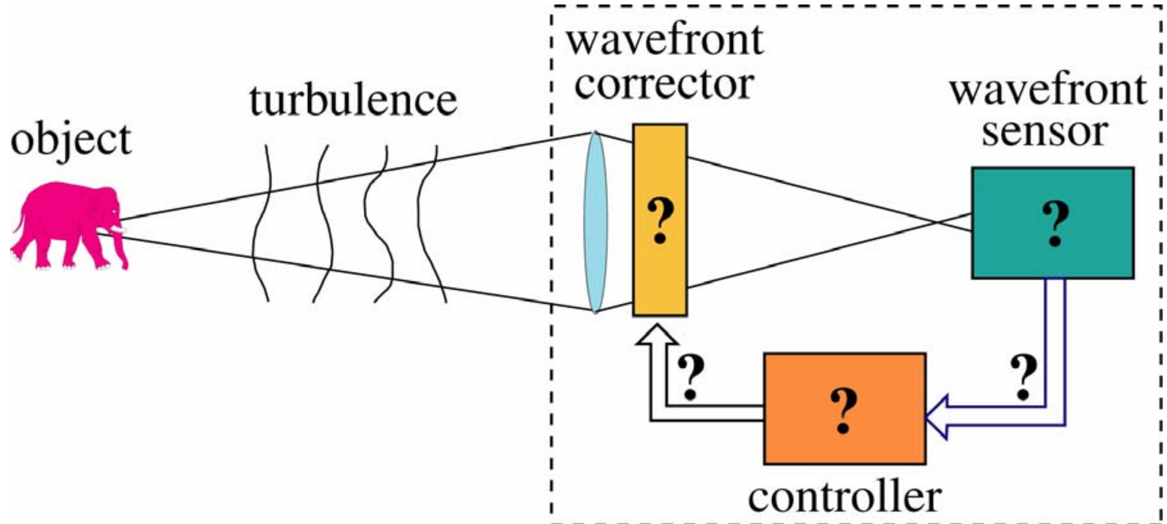
64 Reinforcement Learning Neurons



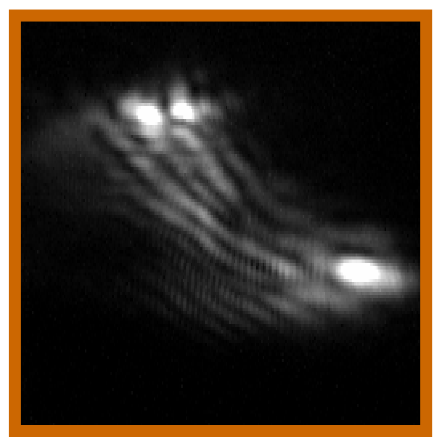
A Biological Adaptive Optics System



Wavefront Distortion and Adaptive Optics

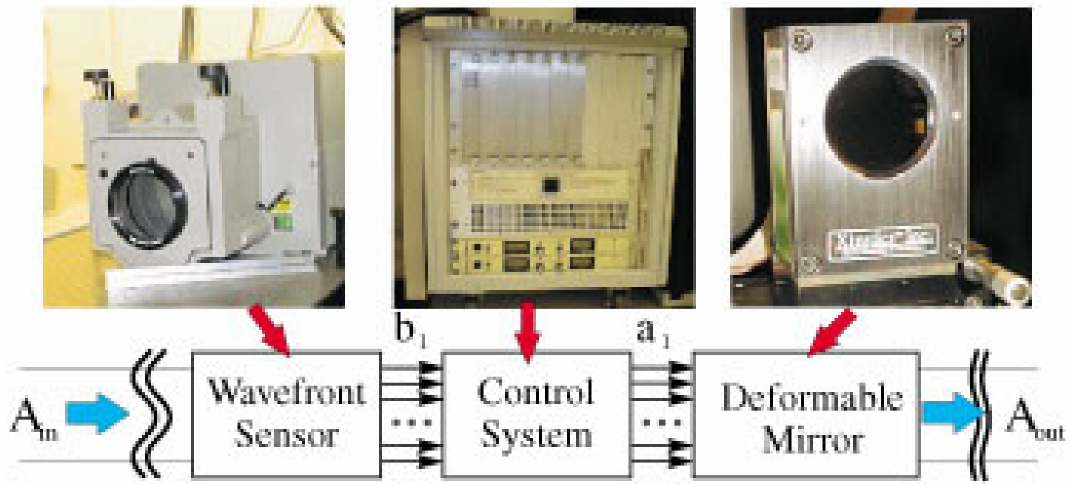


- **Imaging**
  - **defocus**
  - **motion**



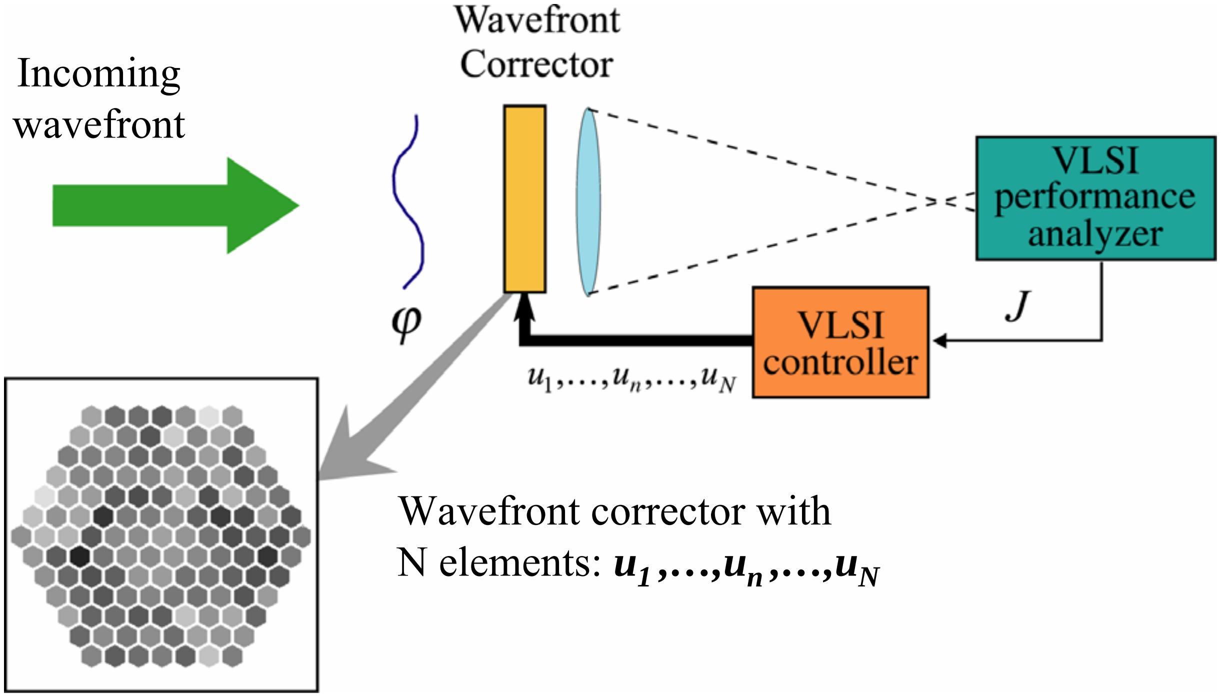
- **Laser beam**
  - **beam wander/spread**
  - **intensity fluctuations**

Adaptive Optics
Conventional Approach



- Performs phase conjugation
  - *assumes intensity is unaffected*
- Complex
  - *requires accurate wavefront phase sensor (Shack-Hartman; Zernike nonlinear filter; etc.)*
  - *computationally intensive control system*

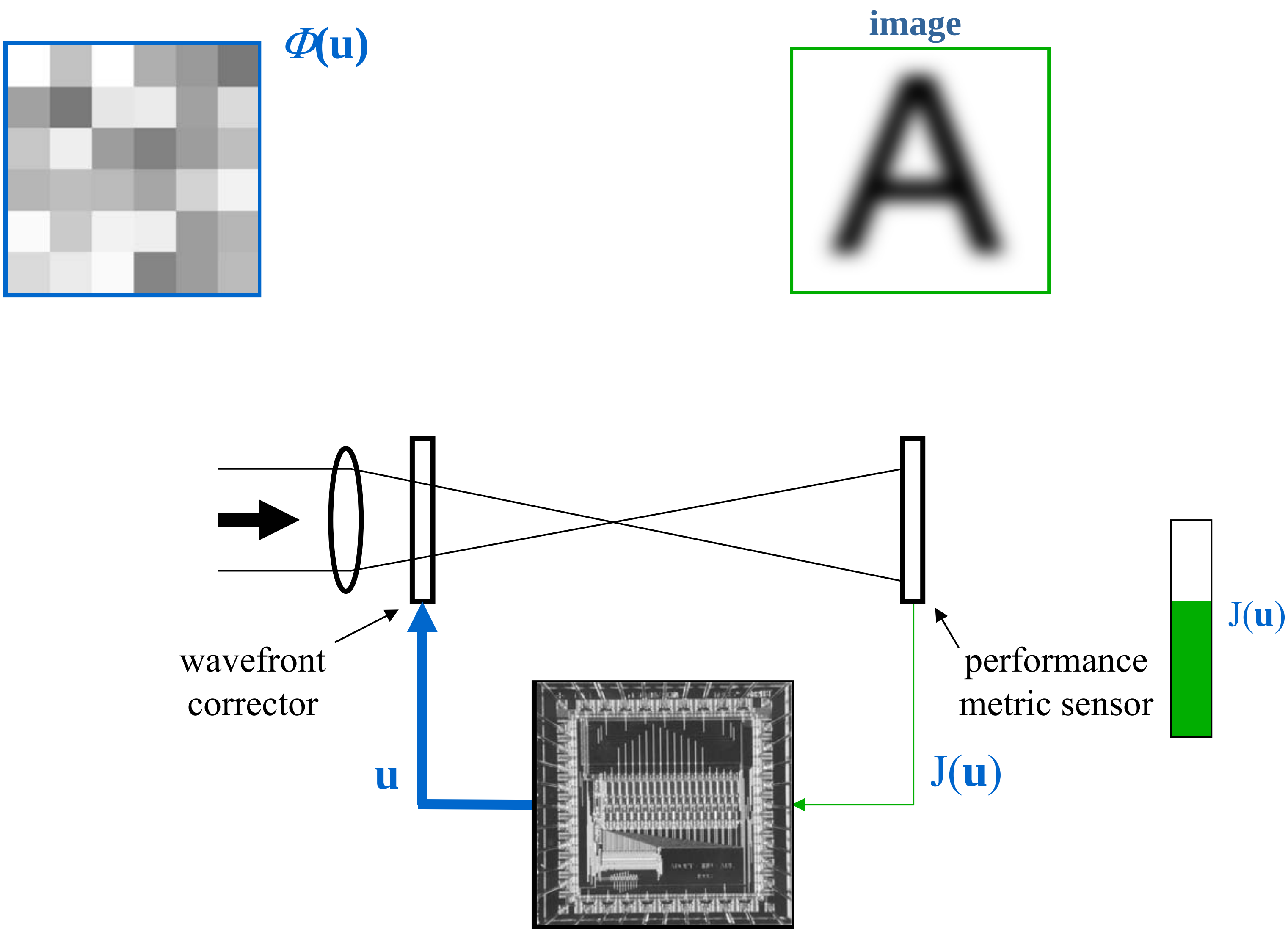
Adaptive Optics
Model-Free Integrated Approach



- Optimizes a direct measure *J* of optical performance ("quality metric")
- No (explicit) model information is required
  - *any type of quality metric J, wavefront corrector (MEMS, LC, …)*
  - *no need for wavefront phase sensor*
- Tolerates imprecision in the implementation of the updates
  - *system level precision limited by accuracy of the measured J*

Adaptive Optics Controller Chip

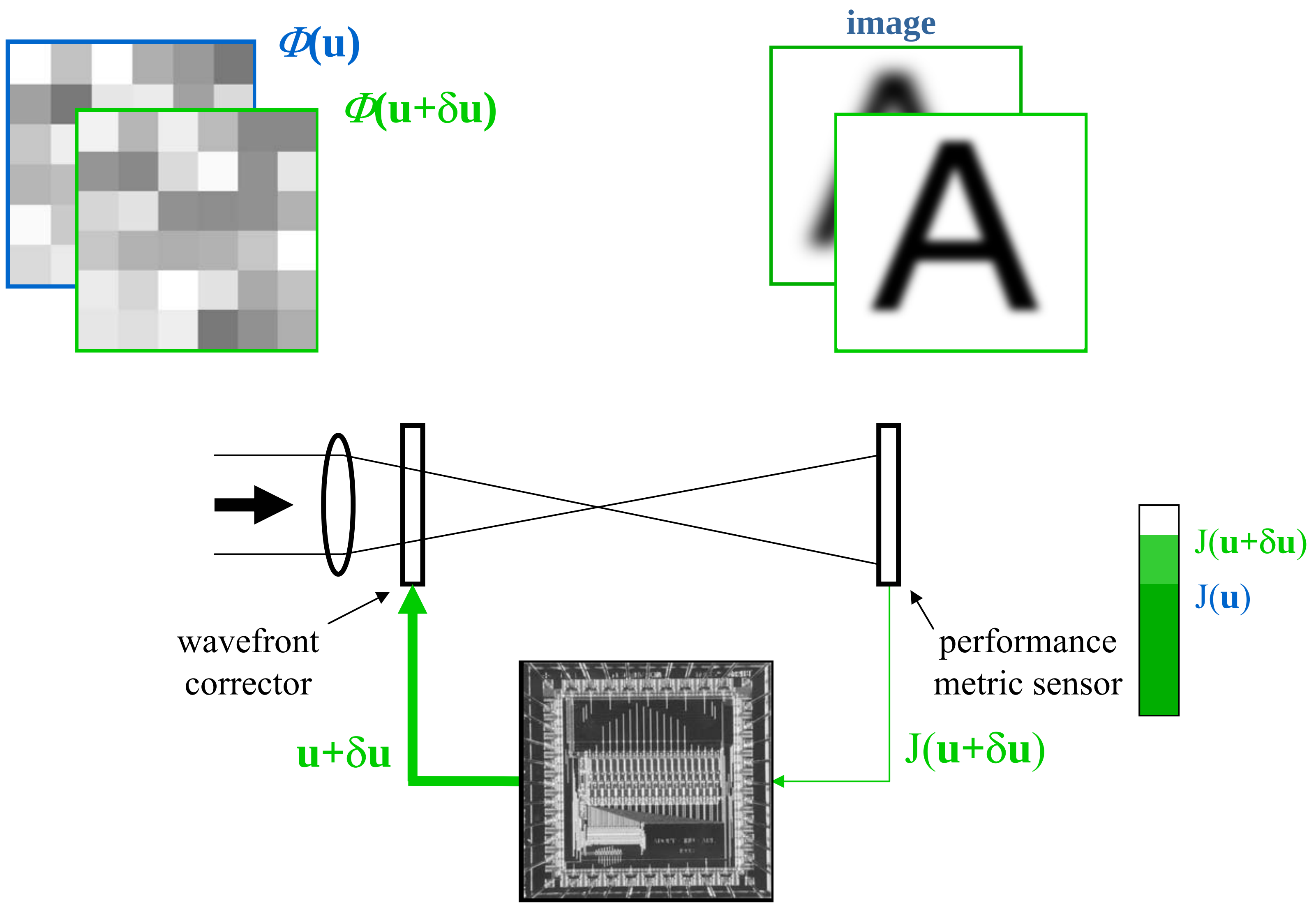
Optimization by Parallel Perturbative Stochastic Gradient Descent



AdOpt VLSI wavefront controller

Adaptive Optics Controller Chip

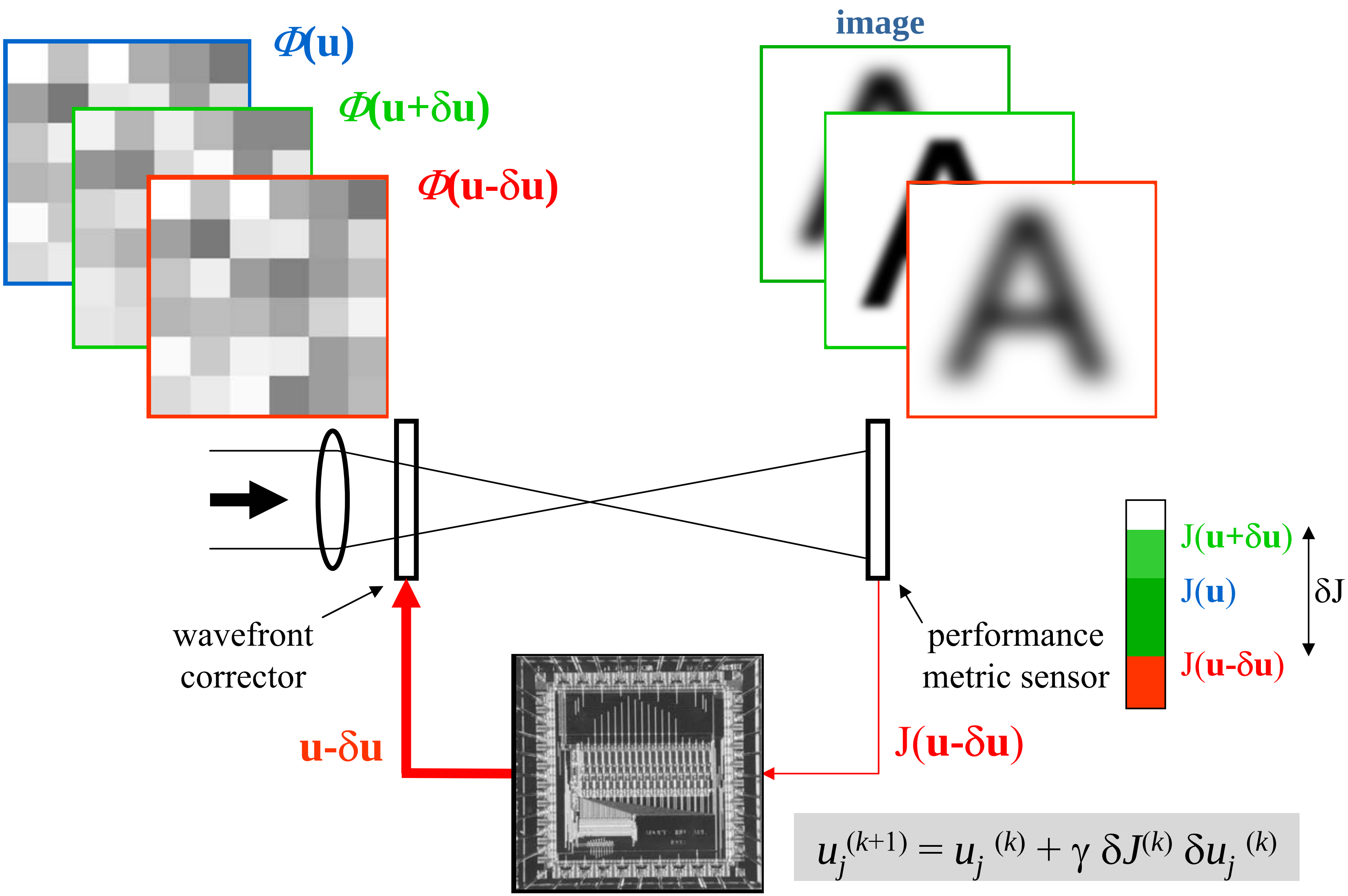
Optimization by Parallel Perturbative Stochastic Gradient Descent



AdOpt VLSI wavefront controller

Adaptive Optics Controller Chip

Optimization by Parallel Perturbative Stochastic Gradient Descent

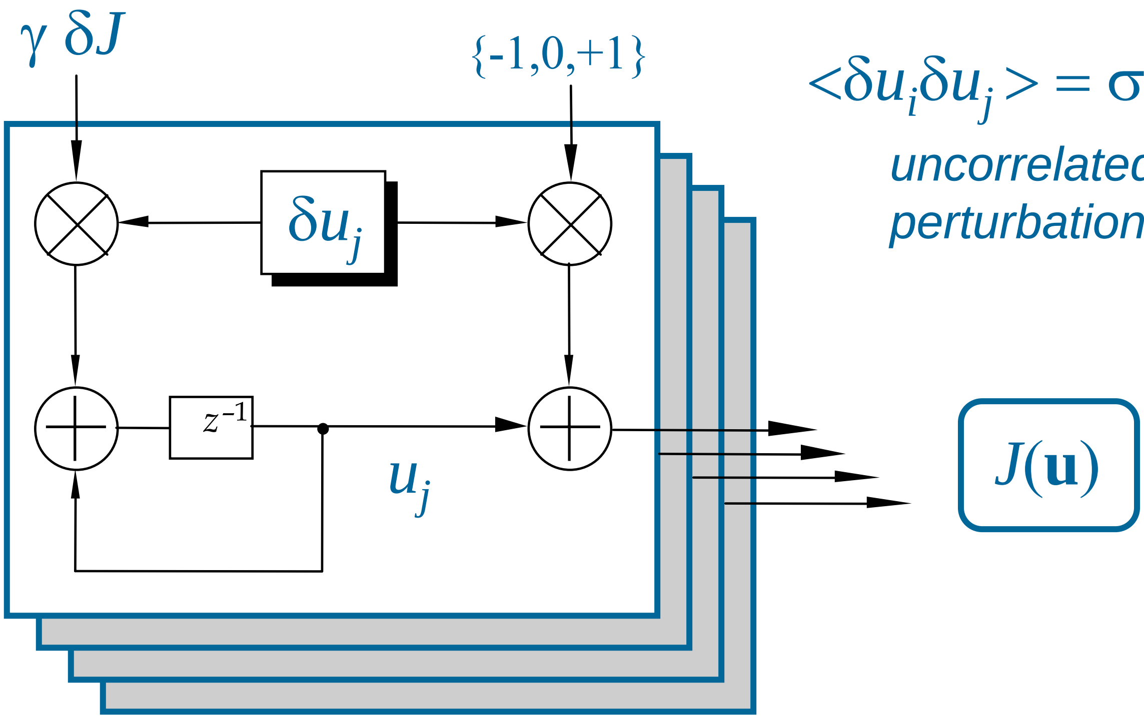


$$u_j^{(k+1)} = u_j^{\ (k)} + \gamma \ \delta J^{(k)} \ \delta u_j^{\ (k)}$$

AdOpt VLSI wavefront controller

Parallel Perturbative Stochastic Gradient Descent

Architecture



$<\delta u_i \delta u_j> = \sigma^2\ \delta_{ij}$

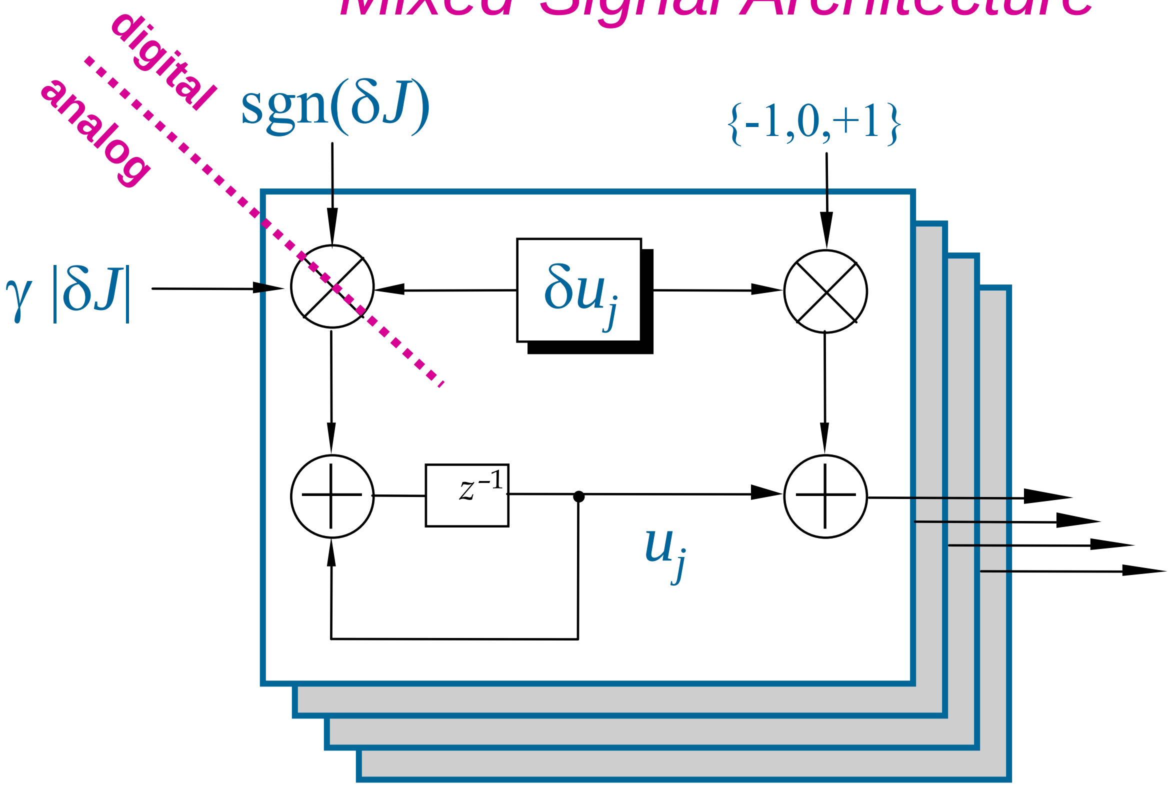
uncorrelated perturbations

$$u_j^{(k+1)} = u_j^{\ (k)} + \gamma\ \delta J^{(k)}\ \delta u_j^{\ (k)}$$

$$2\ \delta J^{(k)} = J\left(u_1^{(k)} + \delta u_1^{\ (k)}, \ldots, u_N^{(k)} + \delta u_N^{\ (k)}\right) - J\left(u_1^{(k)} - \delta u_1^{\ (k)}, \ldots, u_N^{(k)} - \delta u_N^{\ (k)}\right)$$

Parallel Perturbative Stochastic Gradient Descent

Mixed-Signal Architecture



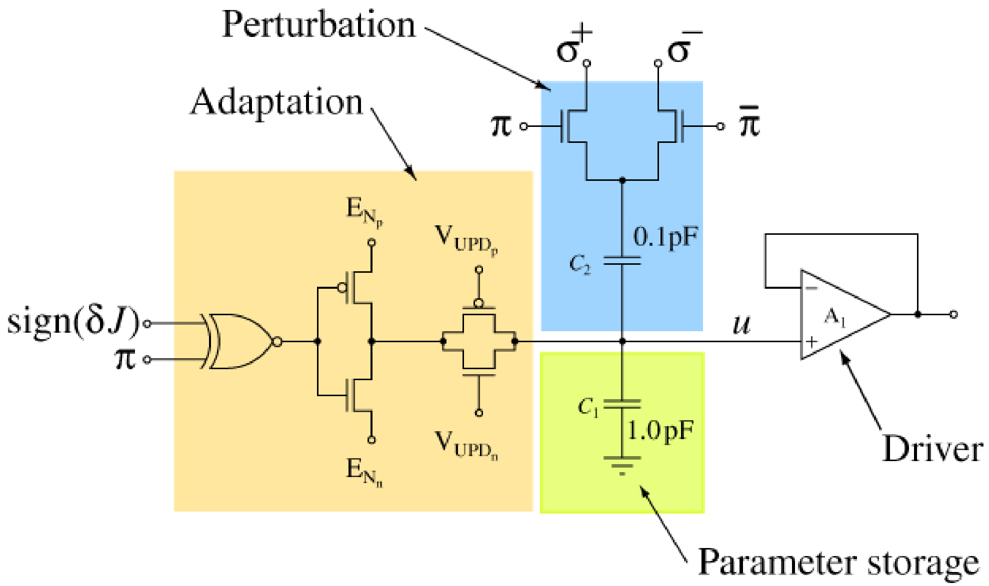
$\delta u_j = \pm 1$

Bernoulli

$$u_j^{(k+1)} = u_j^{(k)} + \underbrace{\gamma\,|\delta J^{(k)}|}_{\text{Amplitude}} \; \underbrace{\mathrm{sgn}(\delta J^{(k)})\, \delta u_j^{(k)}}_{\text{Polarity}}$$

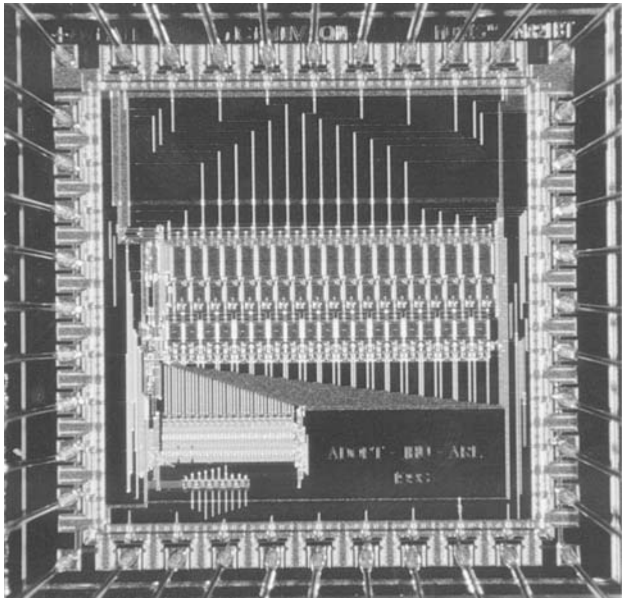
Wavefront Controller
VLSI Implementation
Edwards, Cohen, Cauwenberghs, Vorontsov & Carhart (1999)

- Generate Bernoulli distributed $\left\{\delta u_j^{(k)}\right\}$ with $\left|\delta u_j^{(k)}\right| = \sigma$ and $sgn\left(\delta u_j^{(k)}\right) = \pi_j^{(k)} = \pm 1$
- Decompose $\delta J^{(k)}$ into $sgn\left(\delta J^{(k)}\right) \cdot \left|\delta J^{(k)}\right|$
- Update $u_j^{(k+1)} = u_j^{(k)} - \gamma' \, \mathrm{xor}\left(\mathrm{sgn}\left(\delta J^{(k)}\right) \pi_j^{(k)}\right)$

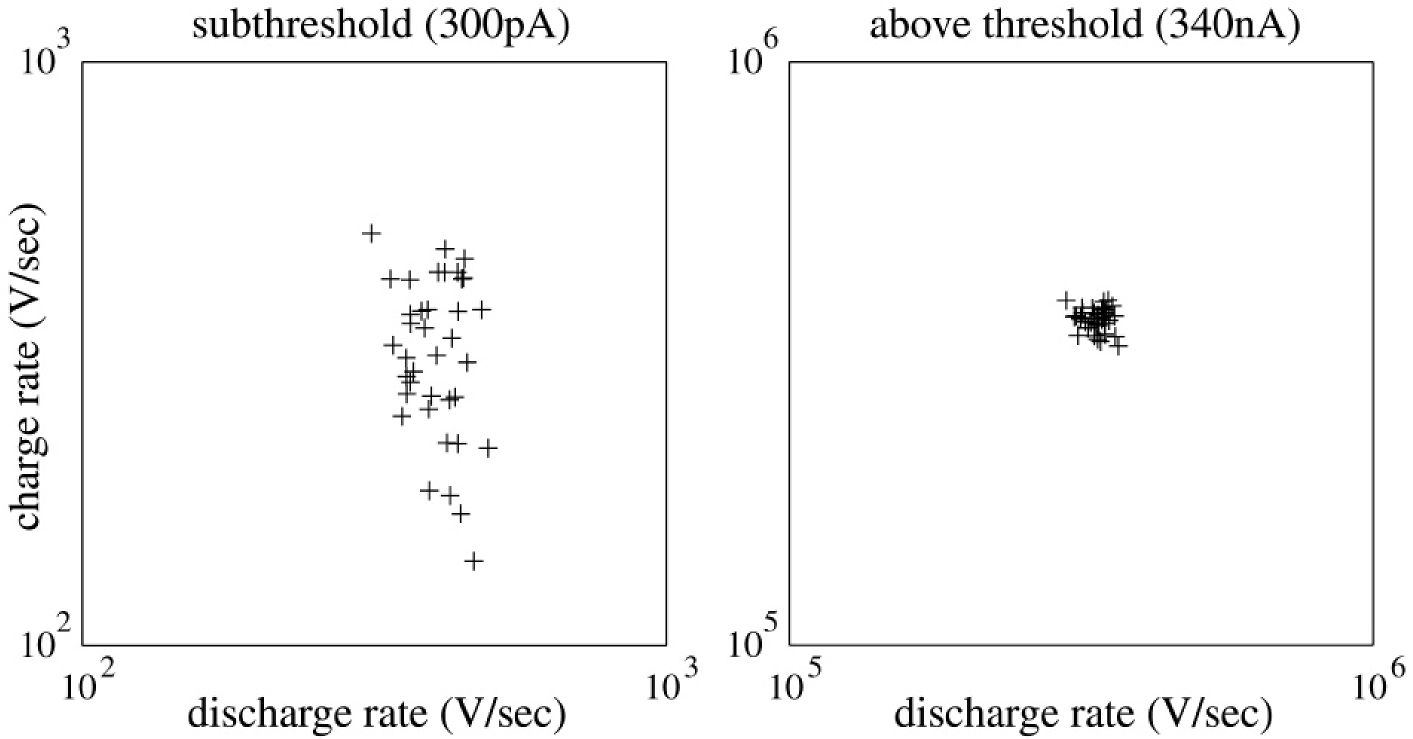


AdOpt mixed-mode chip
2.2 mm^2, 1.2μm CMOS

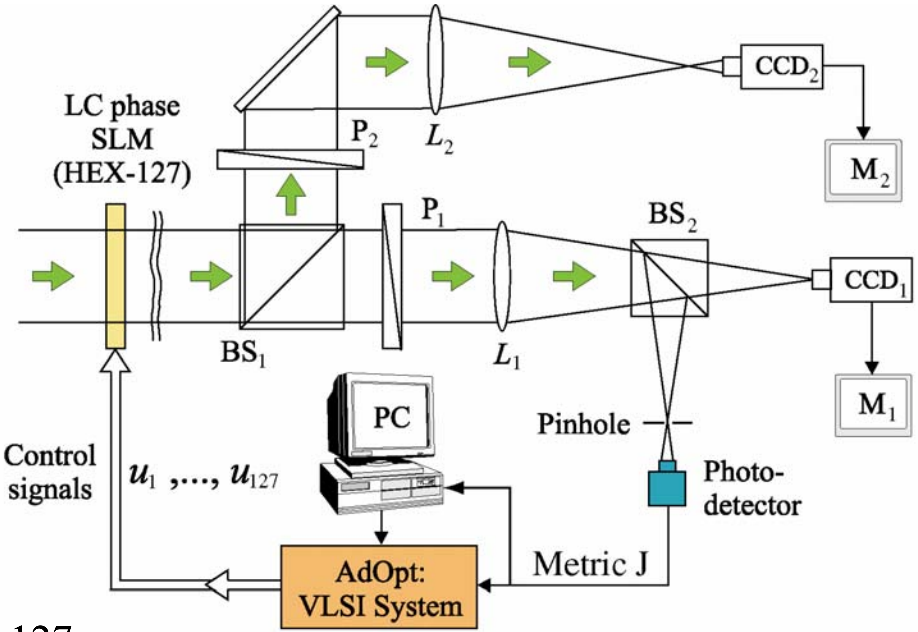
- Controls 19 channels
- Interfaces with LC SLM or MEMS mirrors



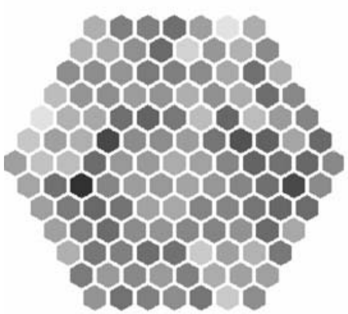
Wavefront Controller
Chip-level Characterization

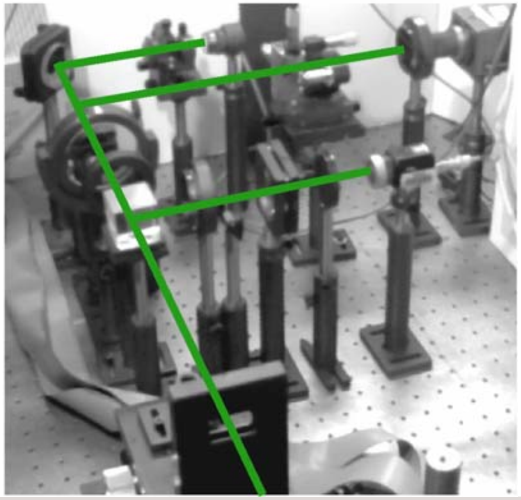


Wavefront Controller
System-level Characterization (LC SLM)



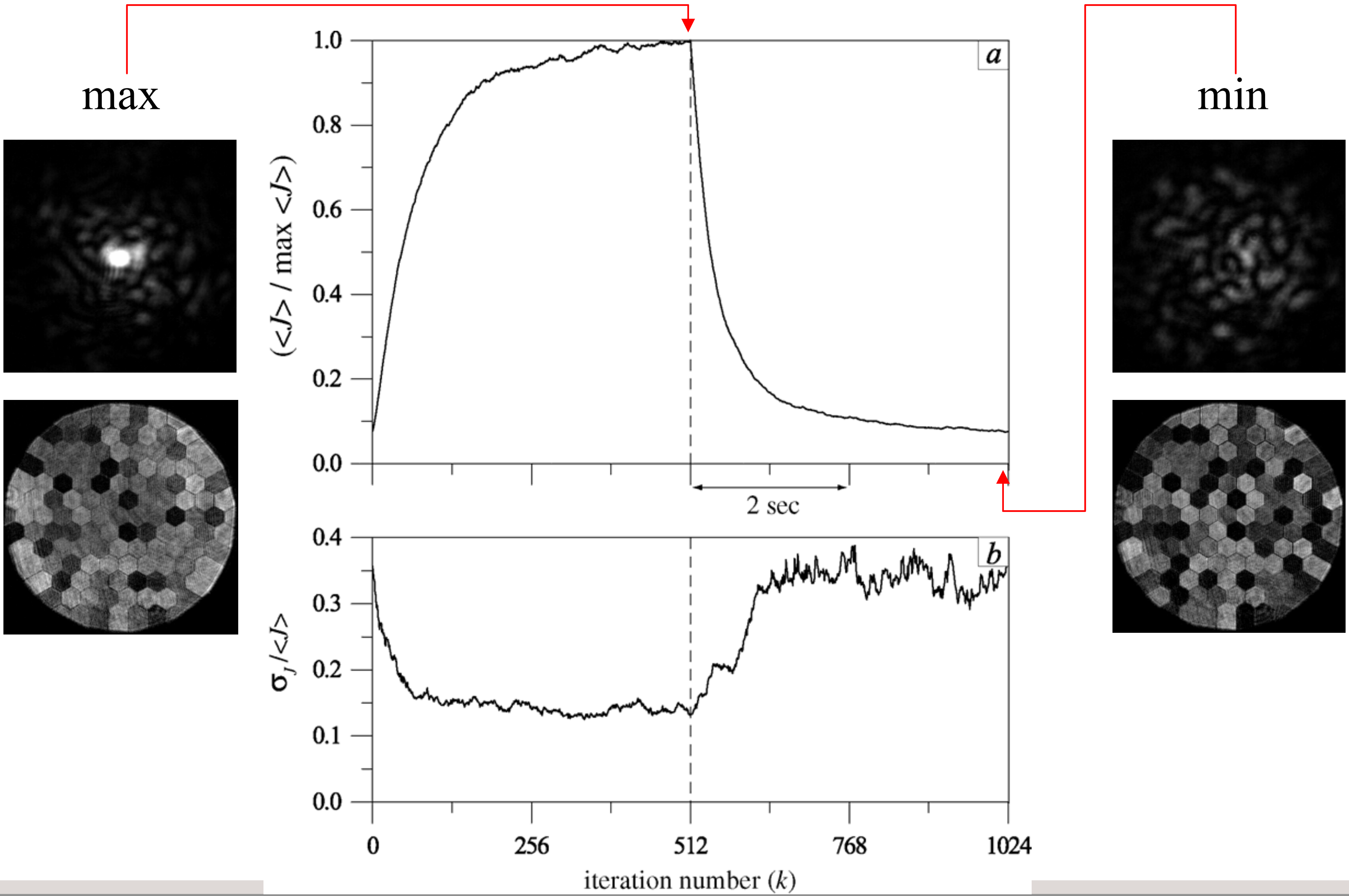
HEX-127 LC SLM





Wavefront Controller
System-level Characterization (SLM)

Maximized and minimized J(**u**) 100 times

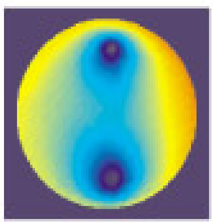


Wavefront Controller

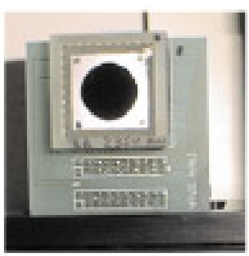
System-Level Characterization (MEMS)

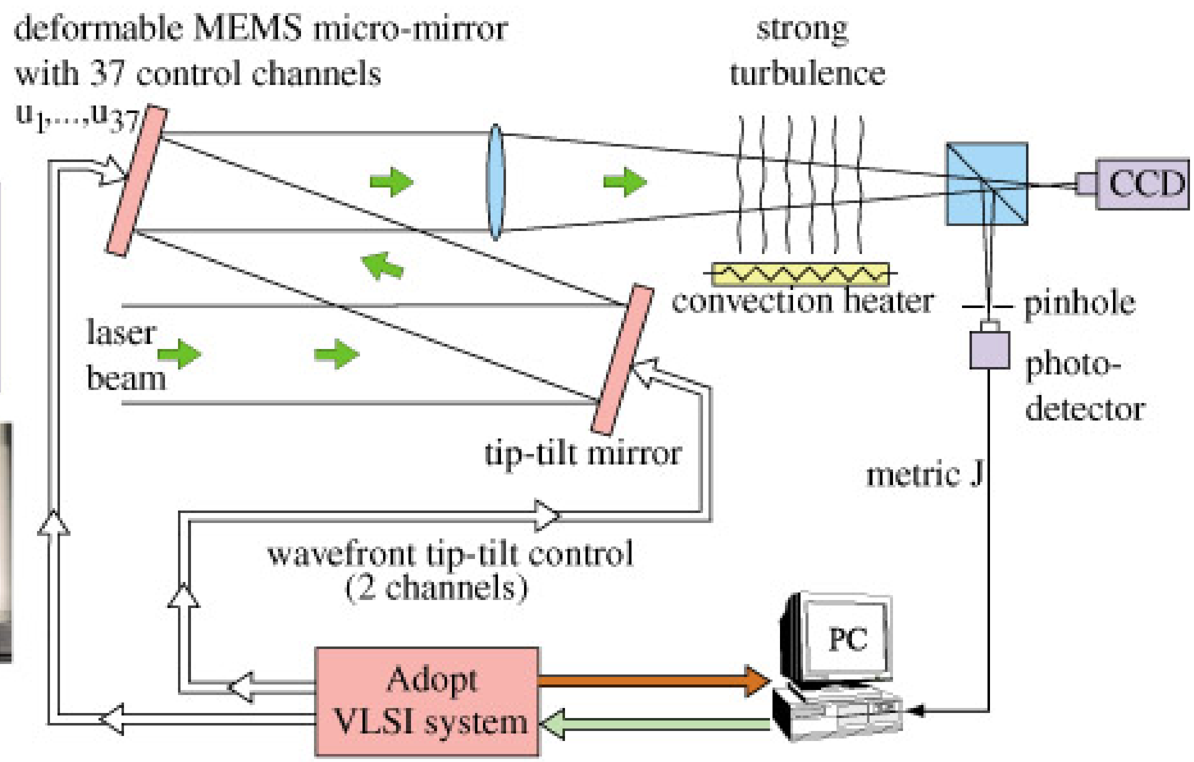
Weyrauch, Vorontsov, Bifano, Hammer, Cohen & Cauwenberghs

Response function



MEMS mirror (OKO)

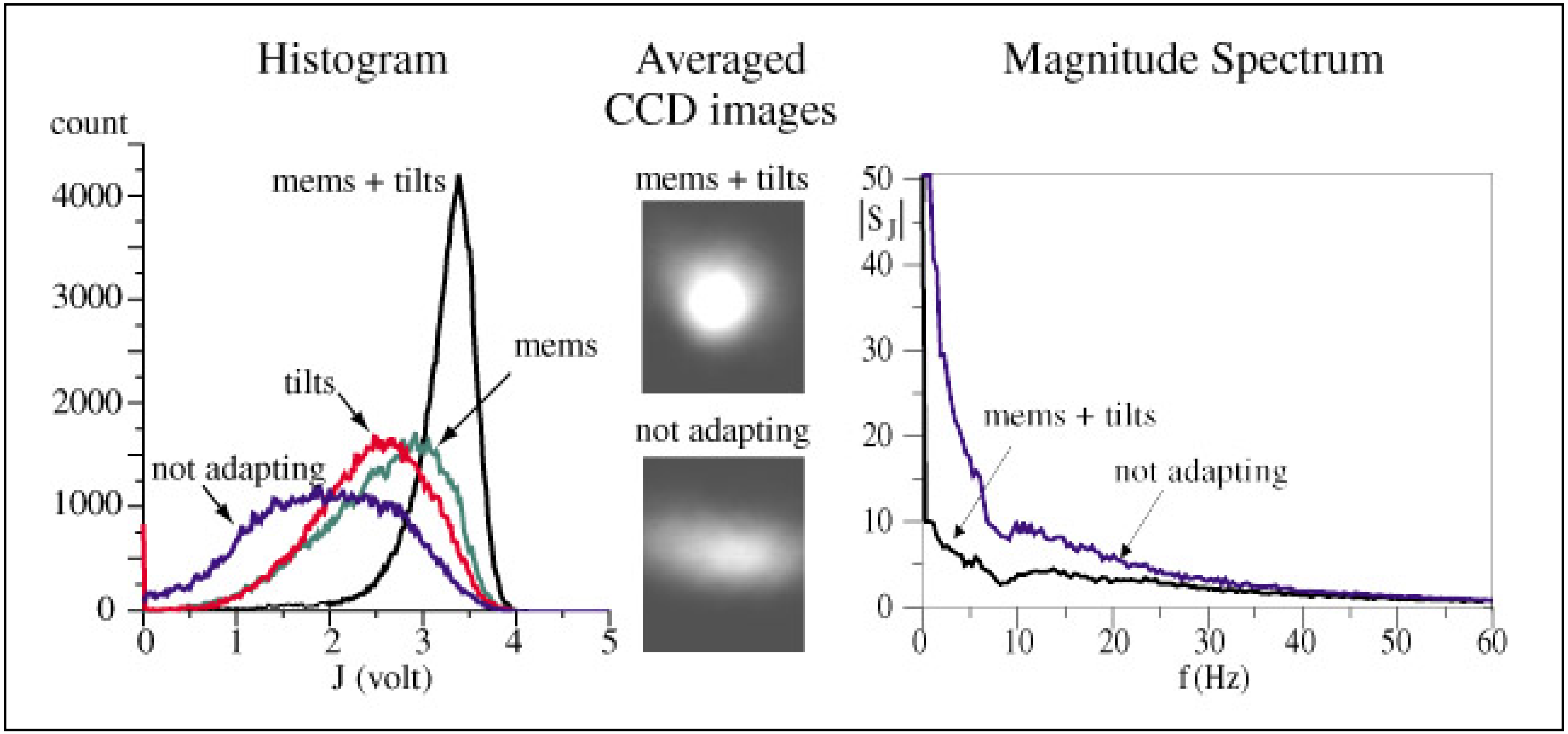




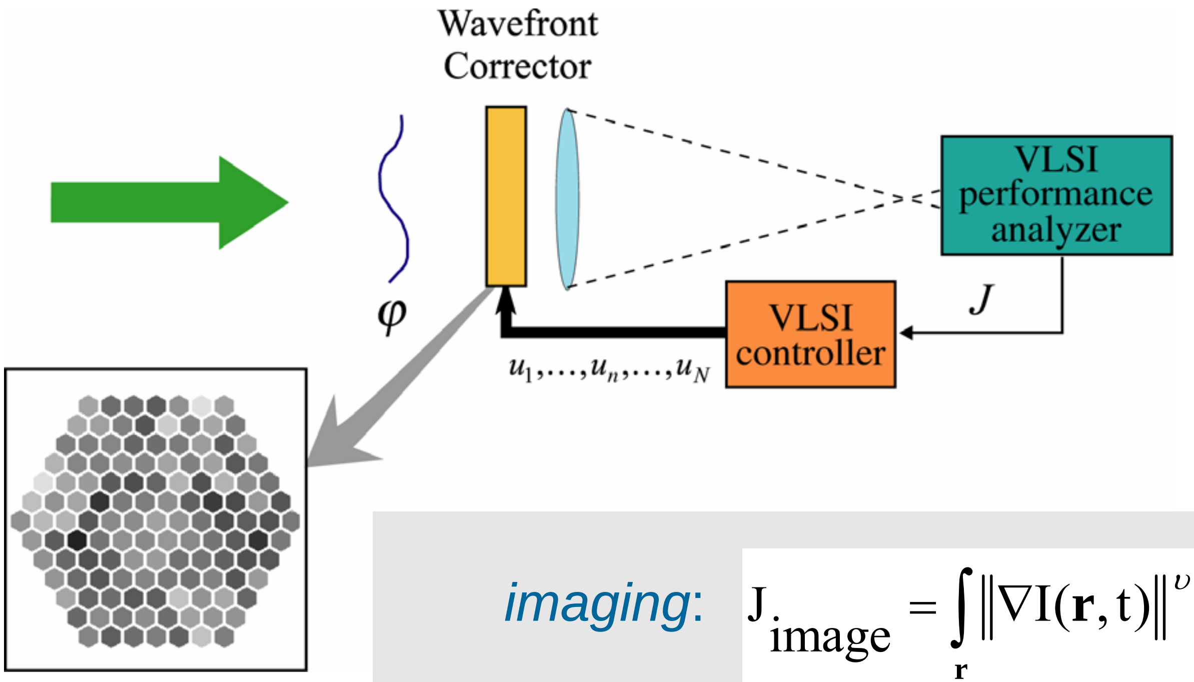
Wavefront Controller

System-level Characterization

(over 500 trials)



Quality Metrics



imaging: $J_{\text{image}} = \int_{\mathbf{r}} \left\| \nabla I(\mathbf{r},t) \right\|^{\upsilon} d^2\mathbf{r}$ — Horn(1968), Delbruck (1999)

laser beam focusing: $J_{\text{beam}} = F\{I(\mathbf{r},t)\}$ — Muller *et al*.(1974) Vorontsov *et al*.(1996)

laser comm: $J_{\text{comm}} = \text{bit-error rate}$

Image Quality Metric Chip

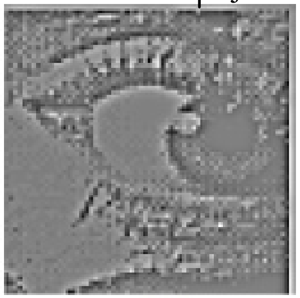
Cohen, Cauwenberghs, Vorontsov & Carhart (2001)

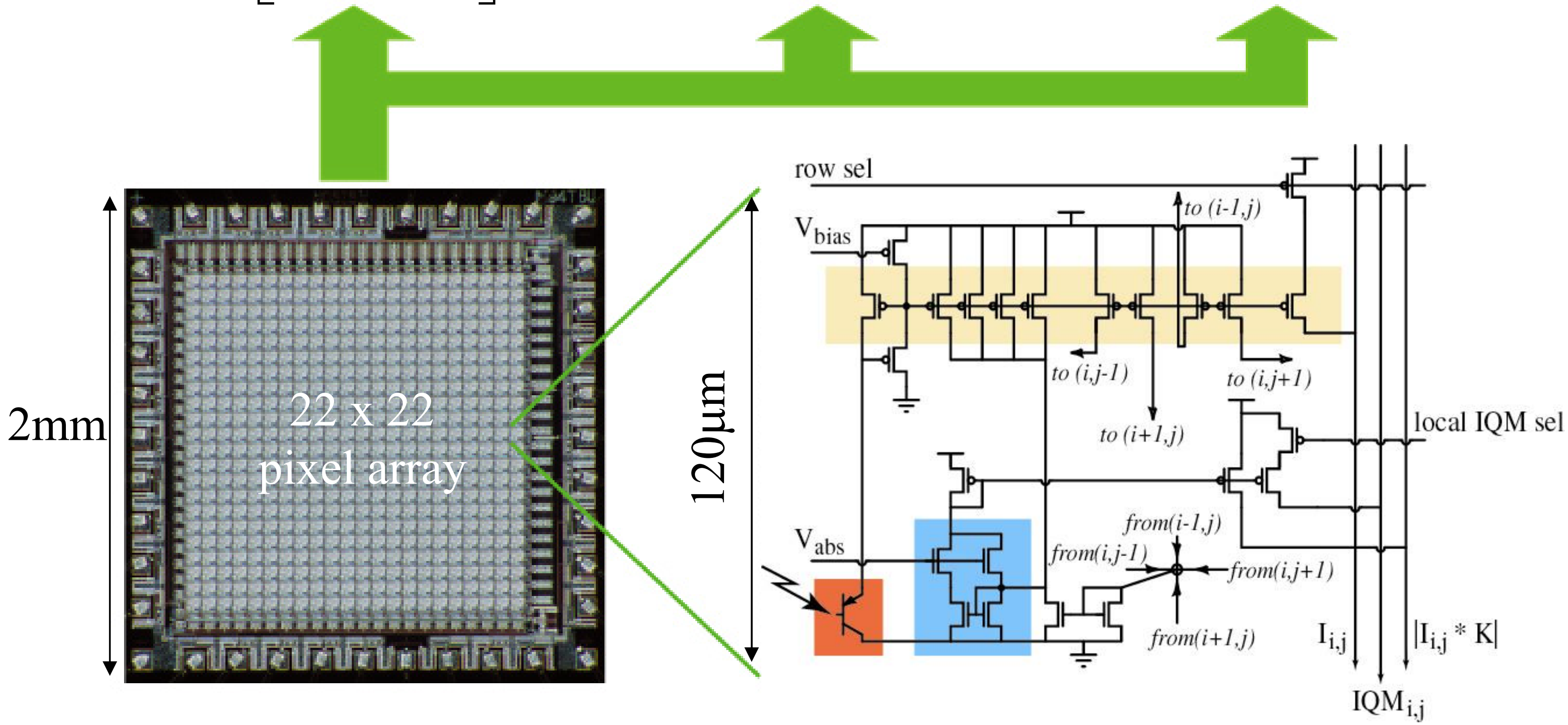
$$\mathrm{IQM} = \sum_{i,j}^{N,M} \left| I_{i,j} * K \right| \Big/ \sum_{i,j}^{N,M} I_{i,j}$$

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

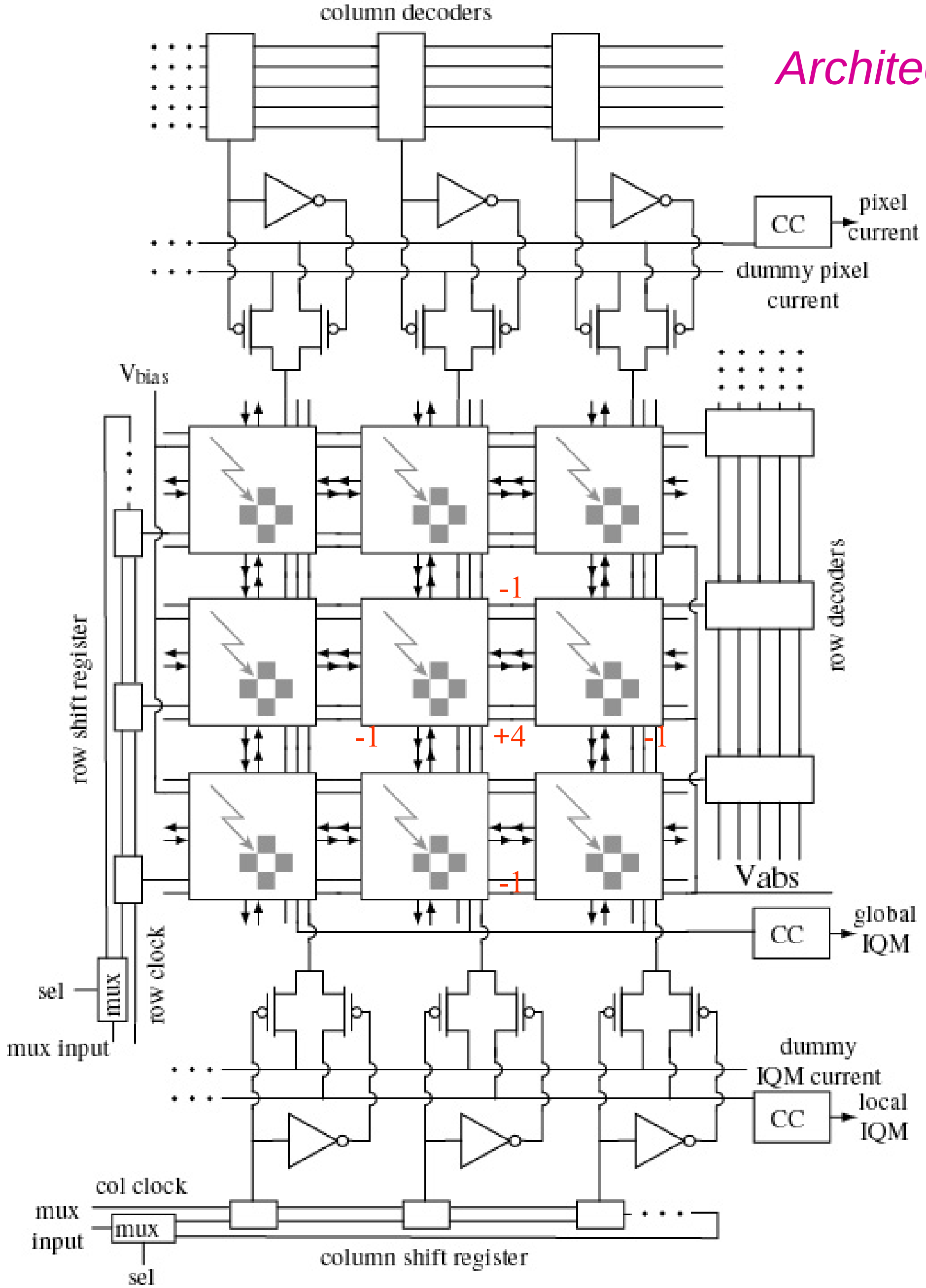
image I_{ij}

edge image $\left| I_{ij} * K \right|$





Architecture (3 x 3)



Edge and Corner Kernels

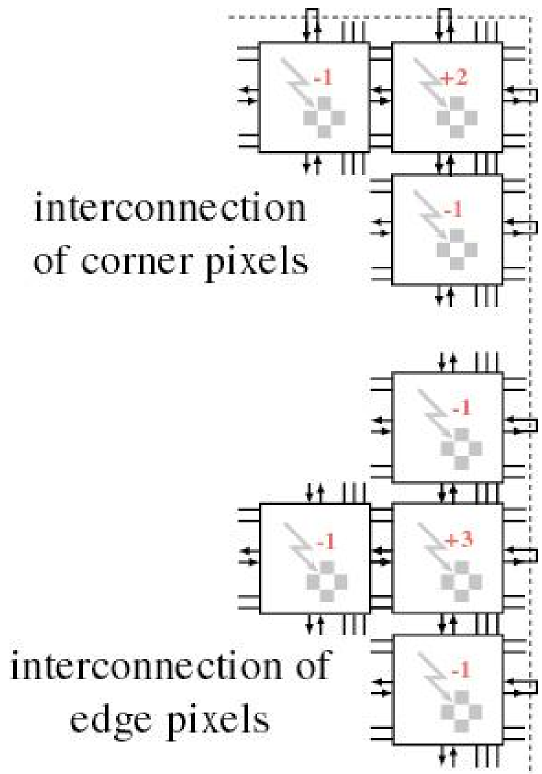


Image Quality Metric Chip Characterization

Experimental Setup

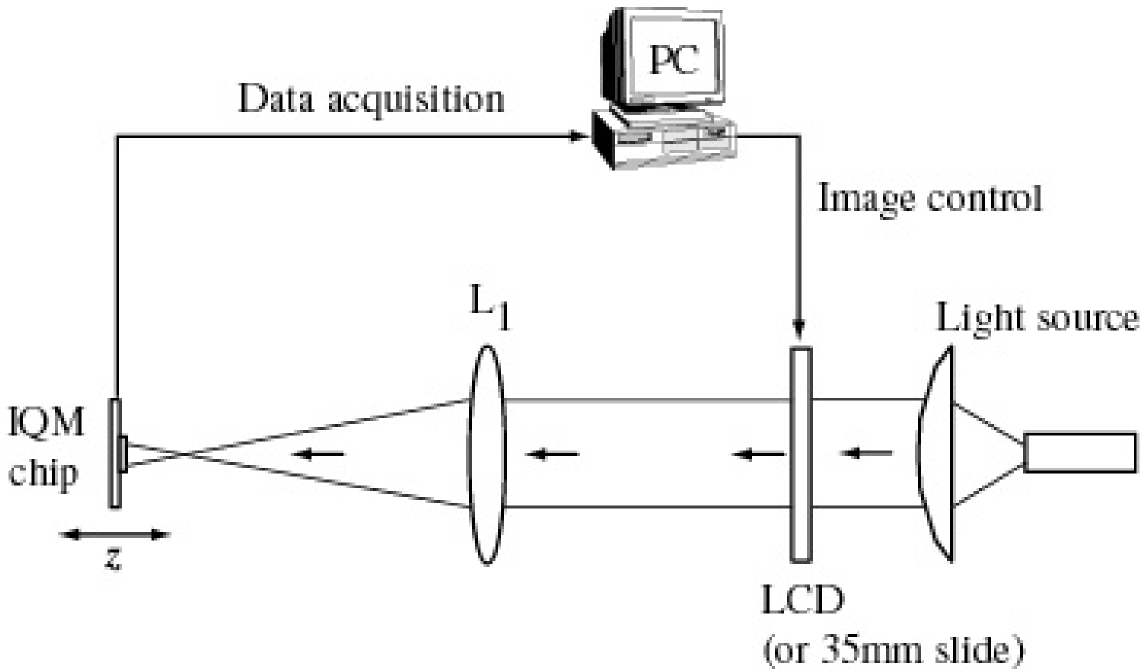


Image Quality Metric Chip Characterization

Experimental Results

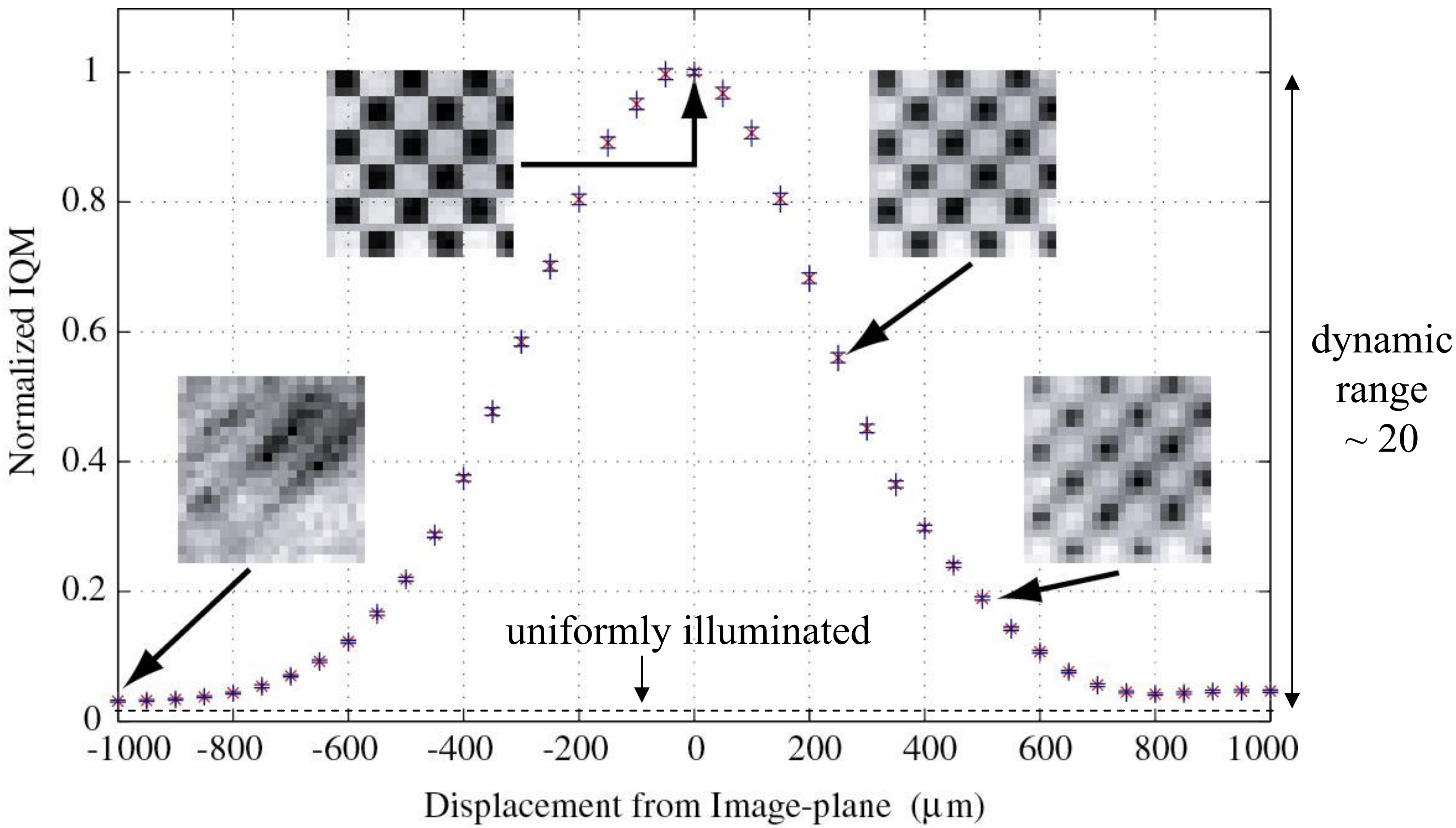
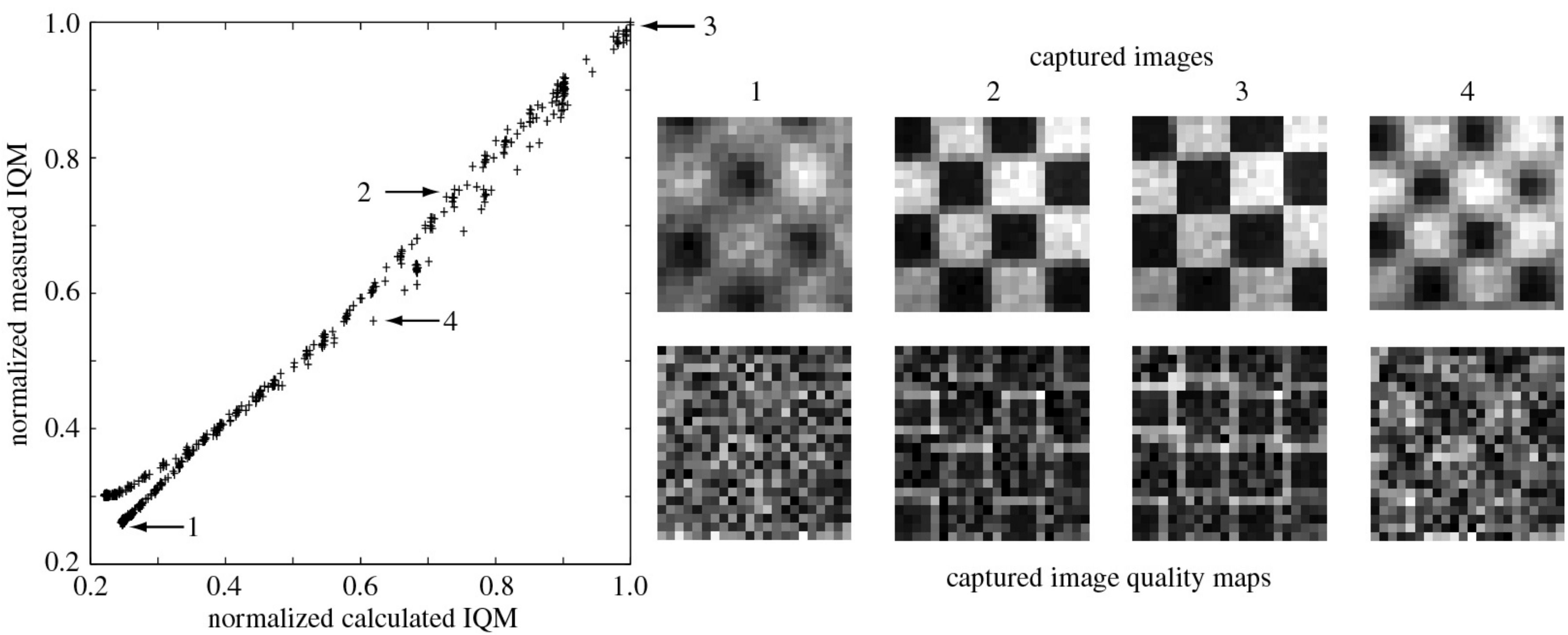


Image Quality Metric Chip Characterization

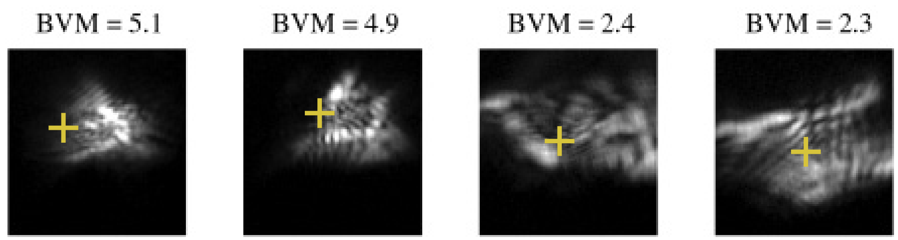
Experimental Results

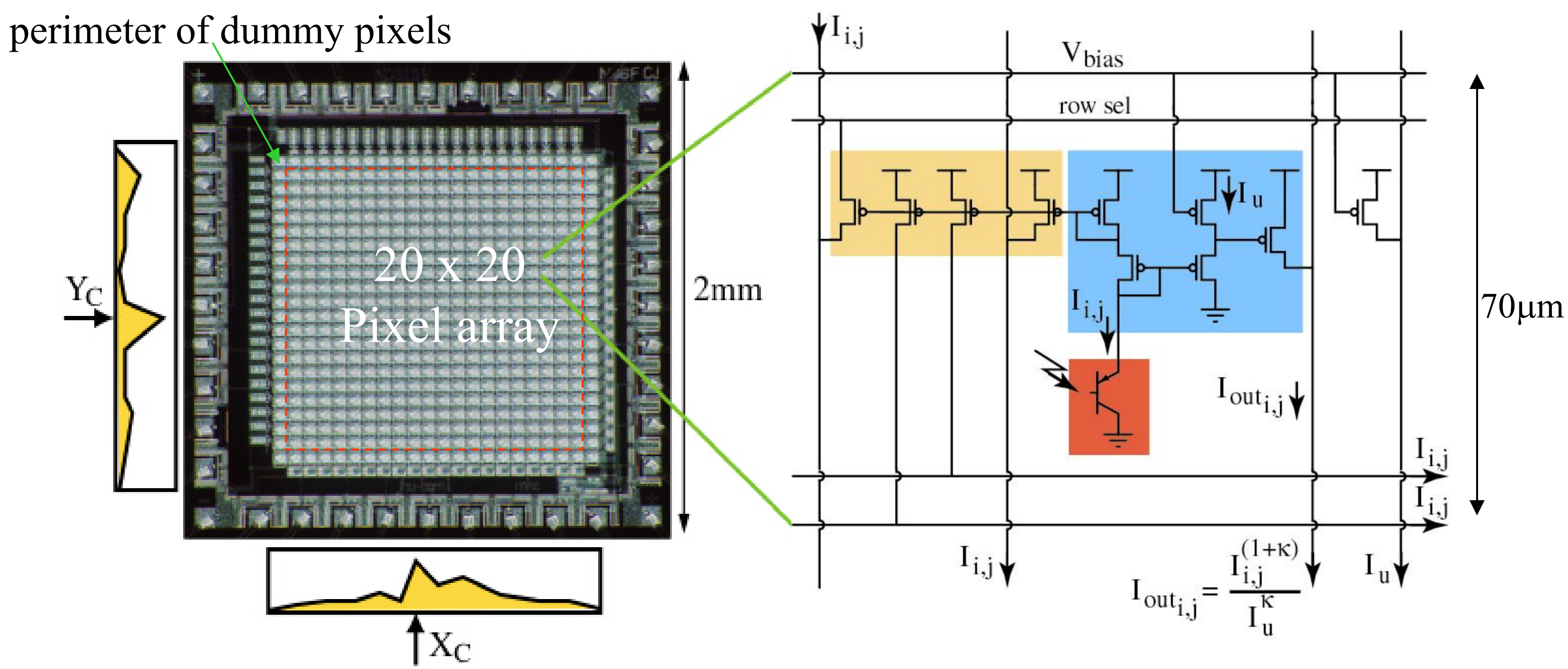


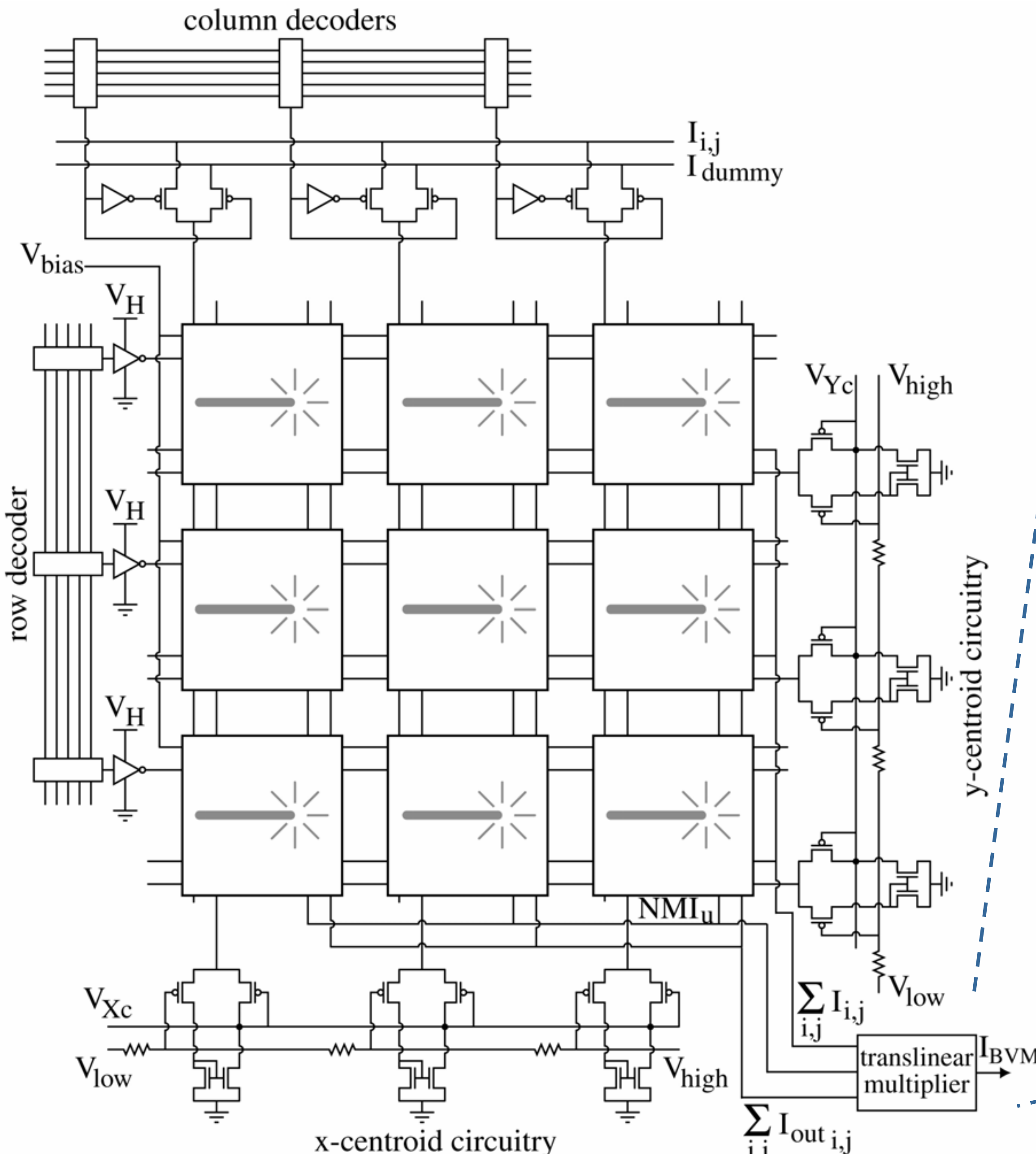
Beam Variance Metric Chip

Cohen, Cauwenberghs, Vorontsov & Carhart (2001)

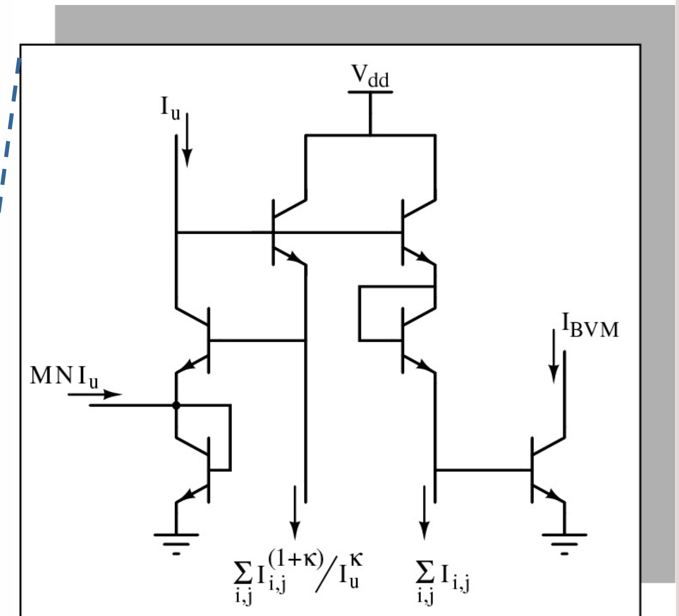
$$\mathrm{BVM} = N \cdot M \cdot \sum_{i,j}^{N,M} I_{i,j}^2 \Big/ \left(\sum_{i,j}^{N,M} I_{i,j} \right)^2$$



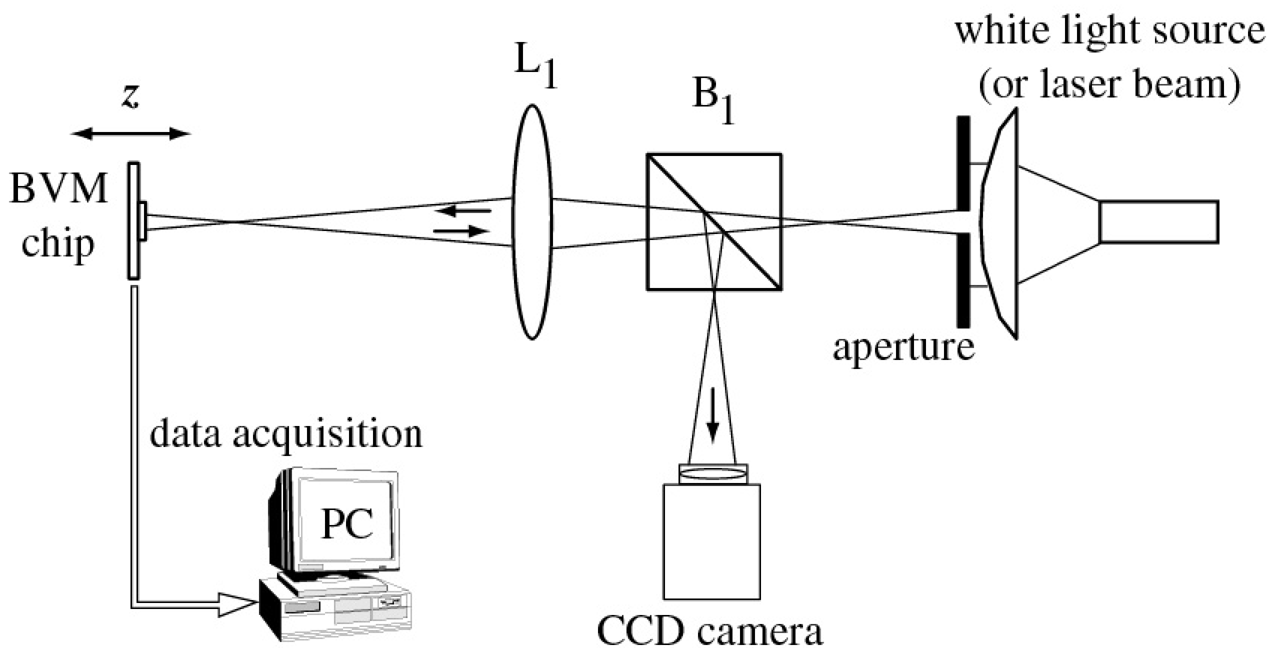




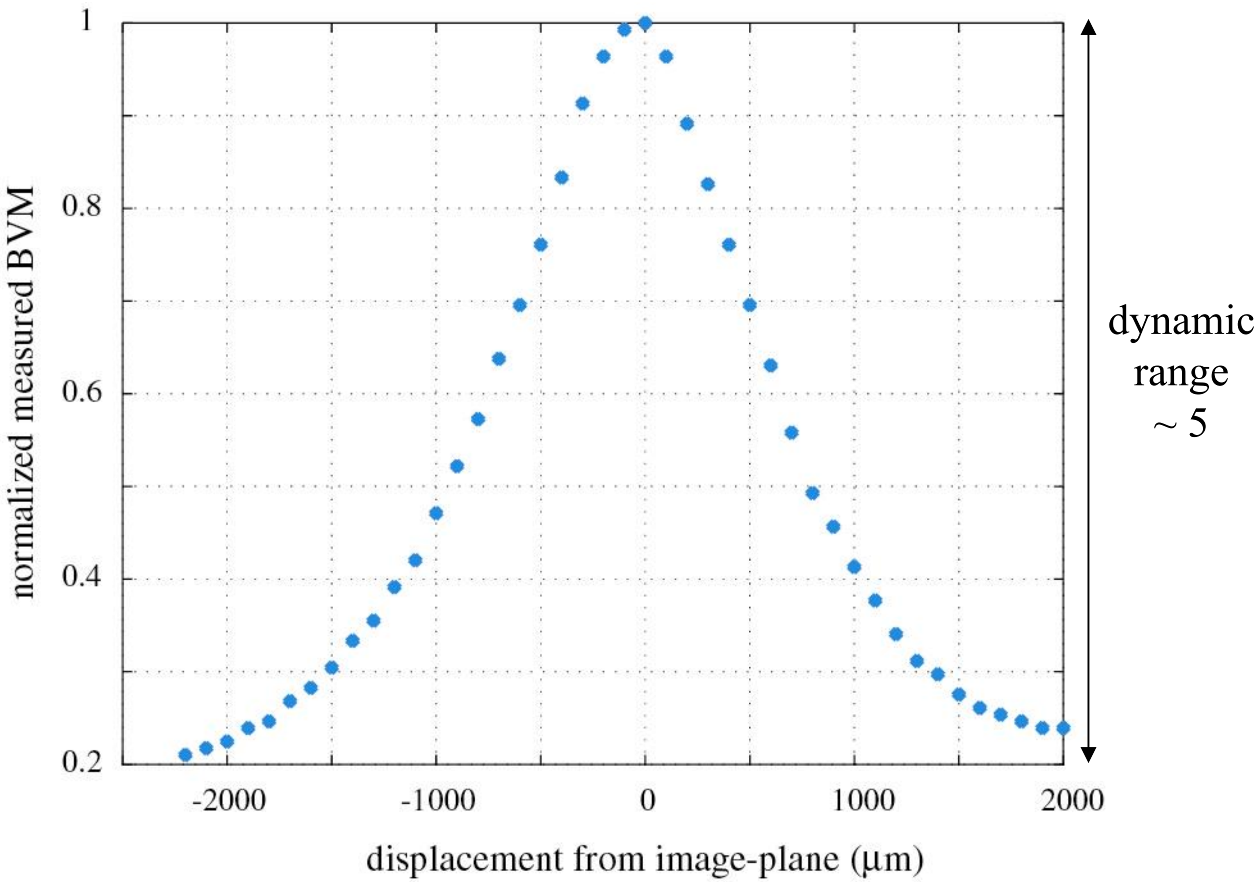
$$I_{BVM} \approx \frac{MNI_u^2 \sum_{i,j}^{N,M} I_{i,j}^{(1+\kappa)} \Big/ I_u^{\kappa}}{\left(\sum_{i,j}^{N,M} I_{i,j}\right)^2}$$



Beam Variance Metric Chip Characterization
Experimental Setup

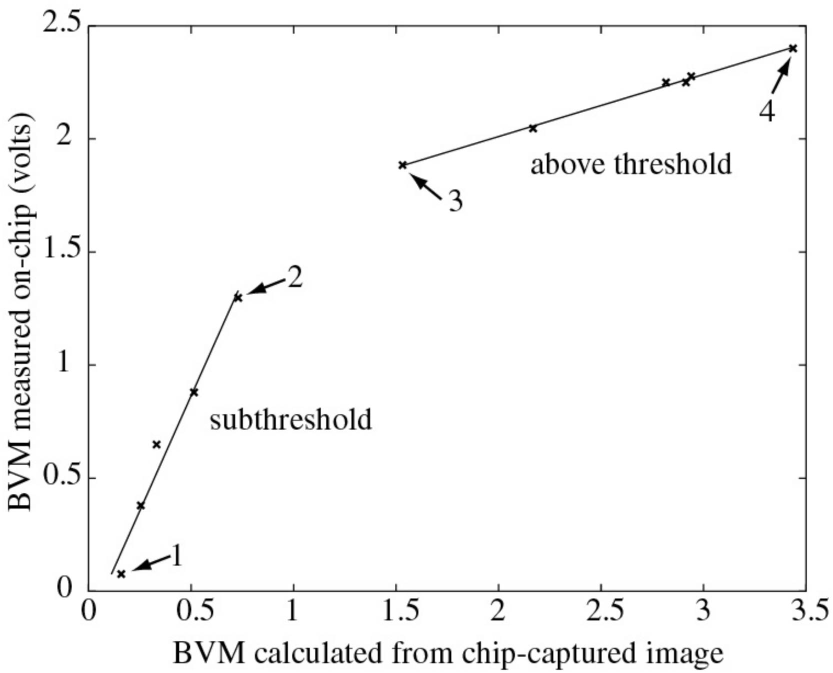


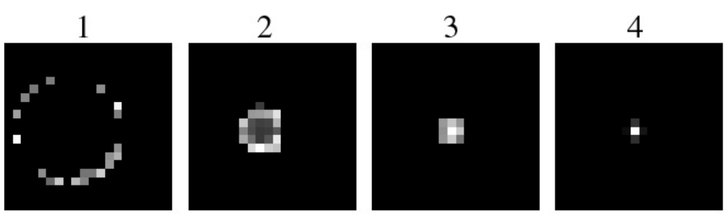
Beam Variance Metric Chip Characterization
Experimental Results



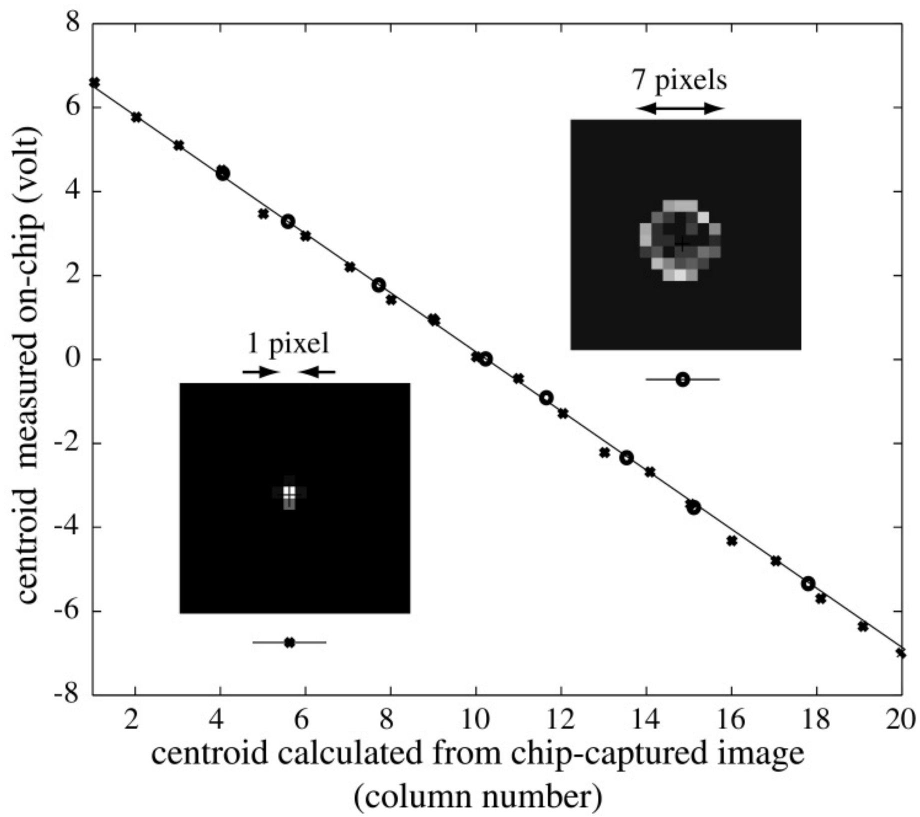
Beam Variance Metric Chip Characterization

Experimental Results



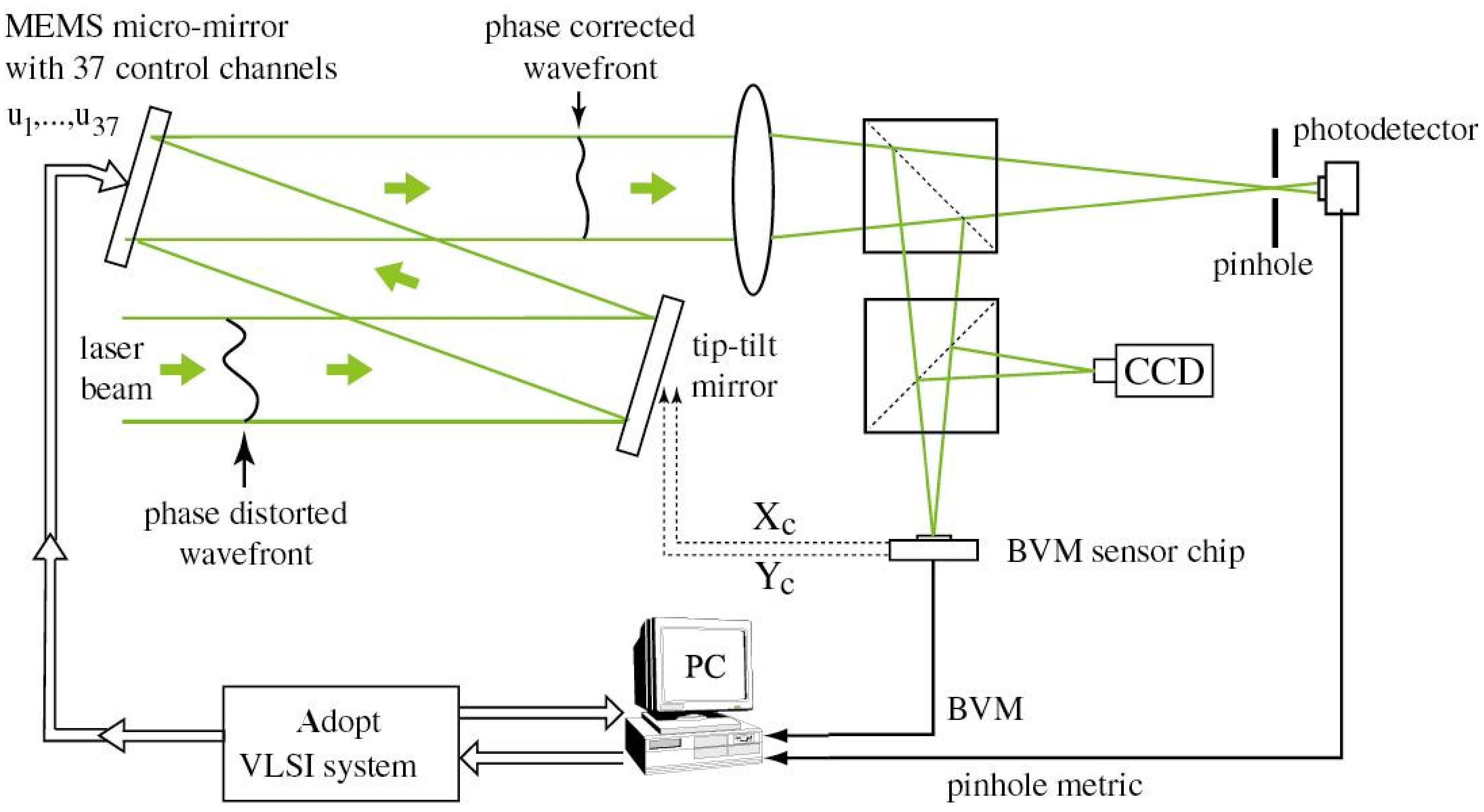


Images from the BVM chip

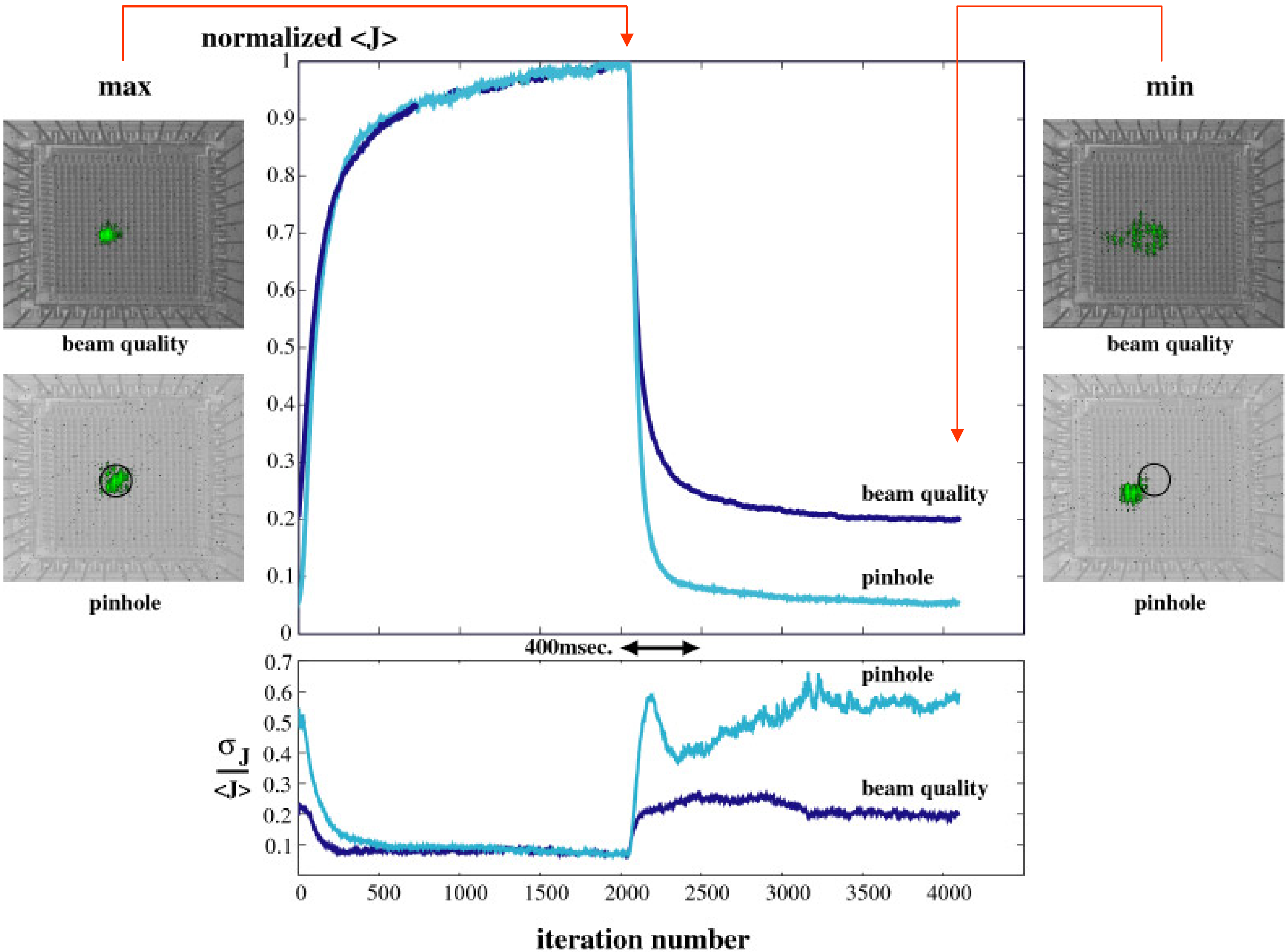


Beam Variance Metric Sensor in the Loop

Laser Receiver Setup



Beam Variance Metric Sensor in the Loop



Conclusions

- *Computational primitives of adaptation and learning are naturally implemented in analog VLSI, and allow to compensate for inaccuracies in the physical implementation of the system under adaptation.*
- *Care should still be taken to avoid inaccuracies in the implementation of the* adaptive element*. Nevertheless, this can easily be achieved by ensuring the correct* polarity*, rather than amplitude, of the parameter update increments.*
- *Adaptation algorithms based on* physical observation *of the "performance" gradient in parameter space are better suited for analog VLSI implementation than algorithms based on a* calculated *gradient.*
- *Among the most generally applicable learning architectures are those that operate on* reinforcement *signals, and those that* blindly *extract and classify signals.*
- *Model-free adaptive optics leads to efficient and robust analog implementation of the control algorithm using a criterion that can be freely chosen to accommodate different wavefront correctors, and different imaging or laser communication applications.*