北京大學 PEKING UNIVERSITY

FPGA Acceleration for Computational Glass-Free Displays

Zhuolun He and Guojie Luo

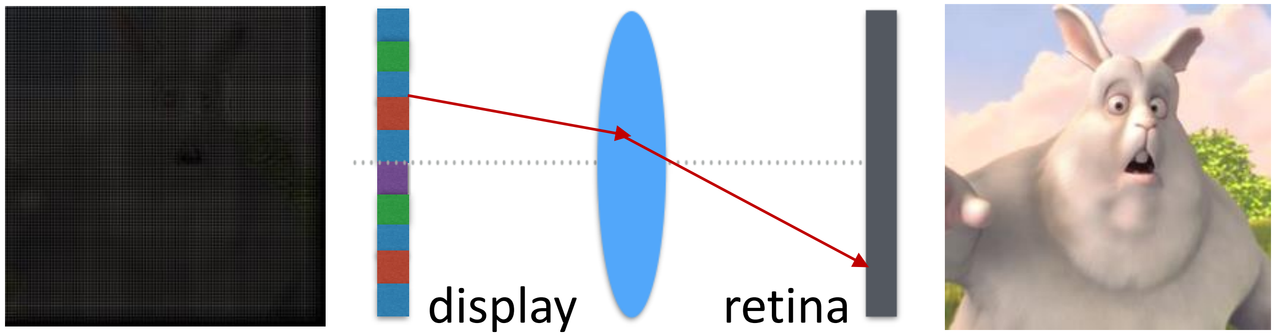
Peking University

FPGA, Feb. 2017

Motivation: hyperopia/myopia Issues

Background Technology: Glass-Free Display

- Light-field display
  - [Huang and Wetzstein, SIGGRAPH 2014]
- Correcting for visual aberrations
  - Display: predistorted content
  - Retina: desired image

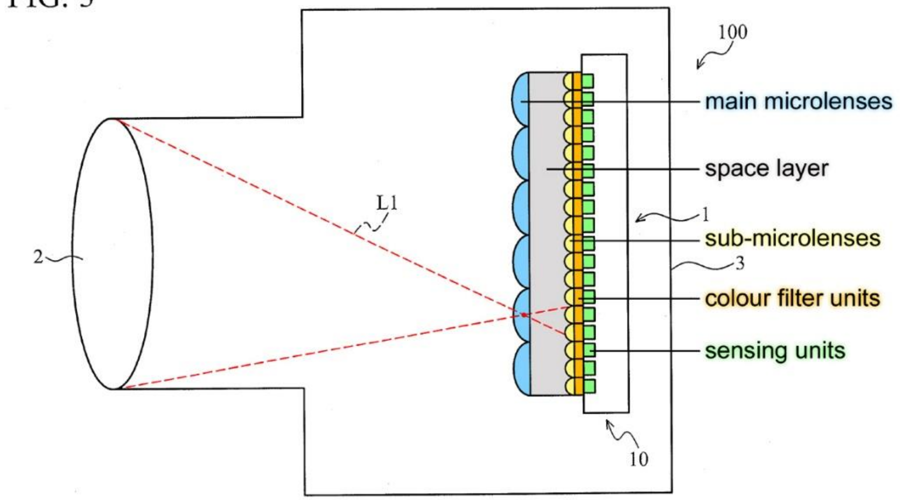


Target light field

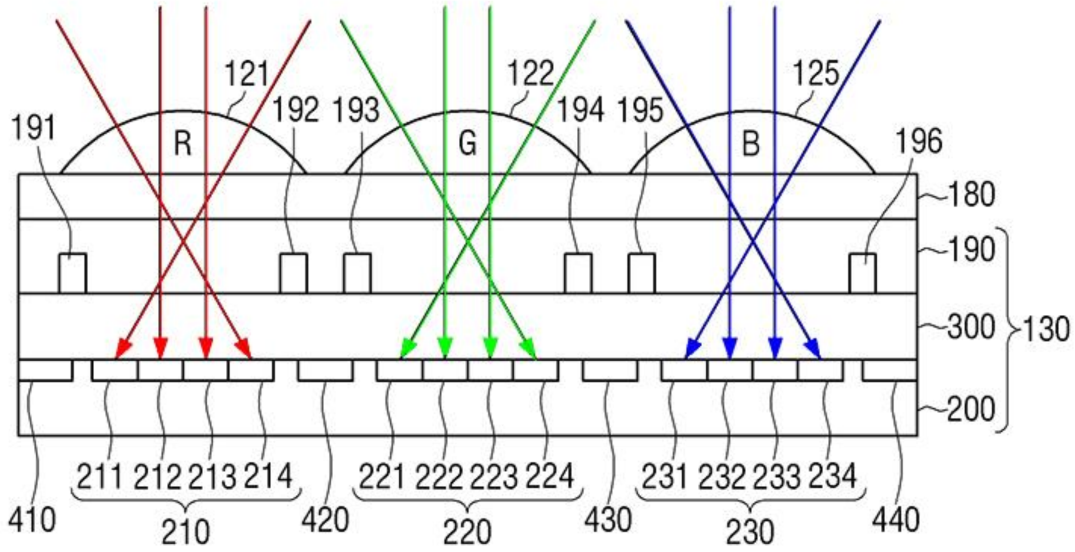
Desired perception

Related Technologies: Light Field Camera

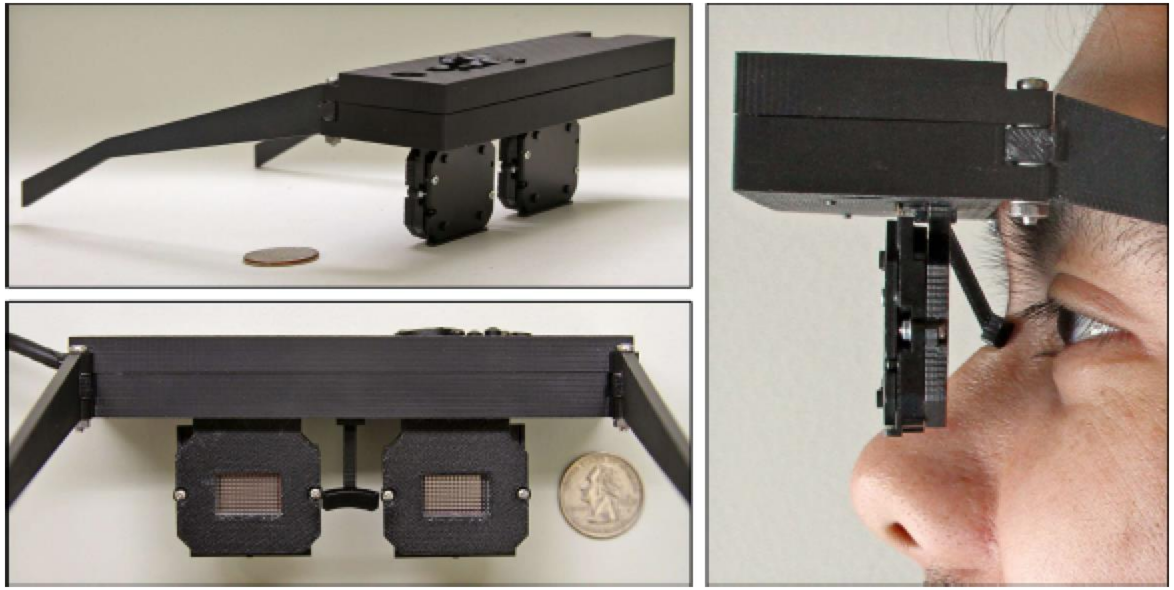
FIG. 3

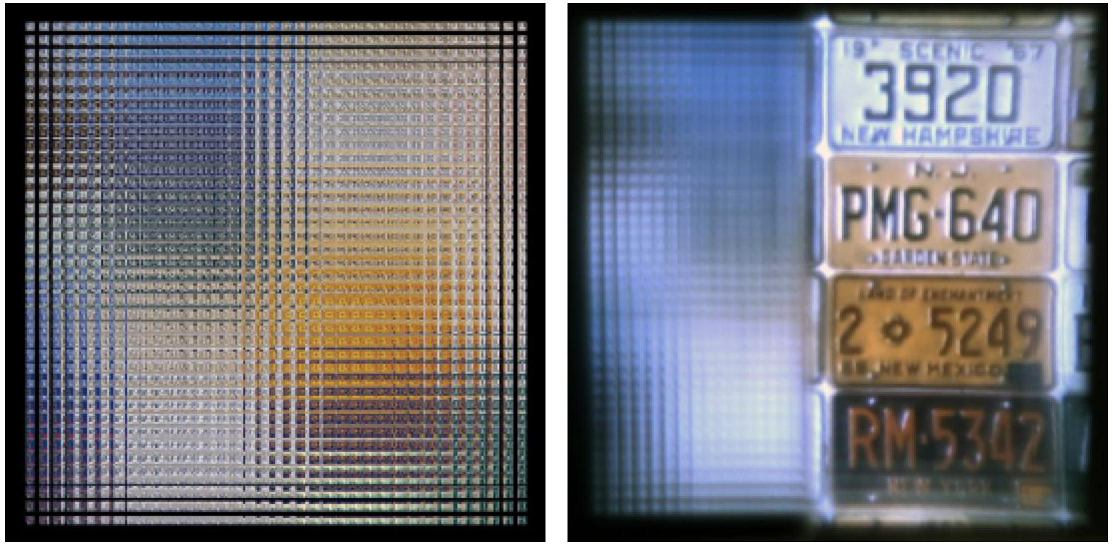


Wang et al. 2014, modified by www.lightfield-forum.com



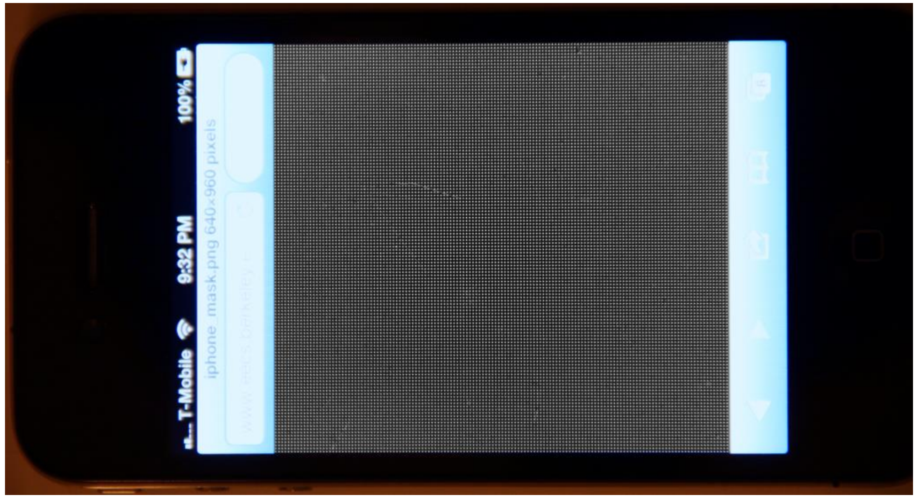
Related Tech.: Near-eye Light-field Display





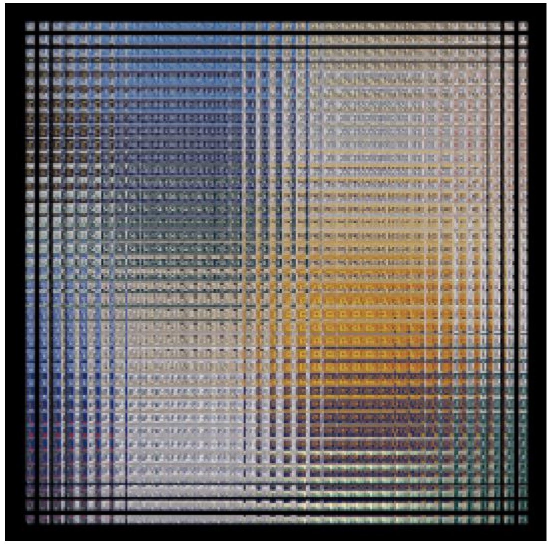
Source: NVIDIA, SIGGRAPH Asia 2013

Pinhole Array vs. Microlens



One 75um pinhole in every 390um
manufactured using lithography

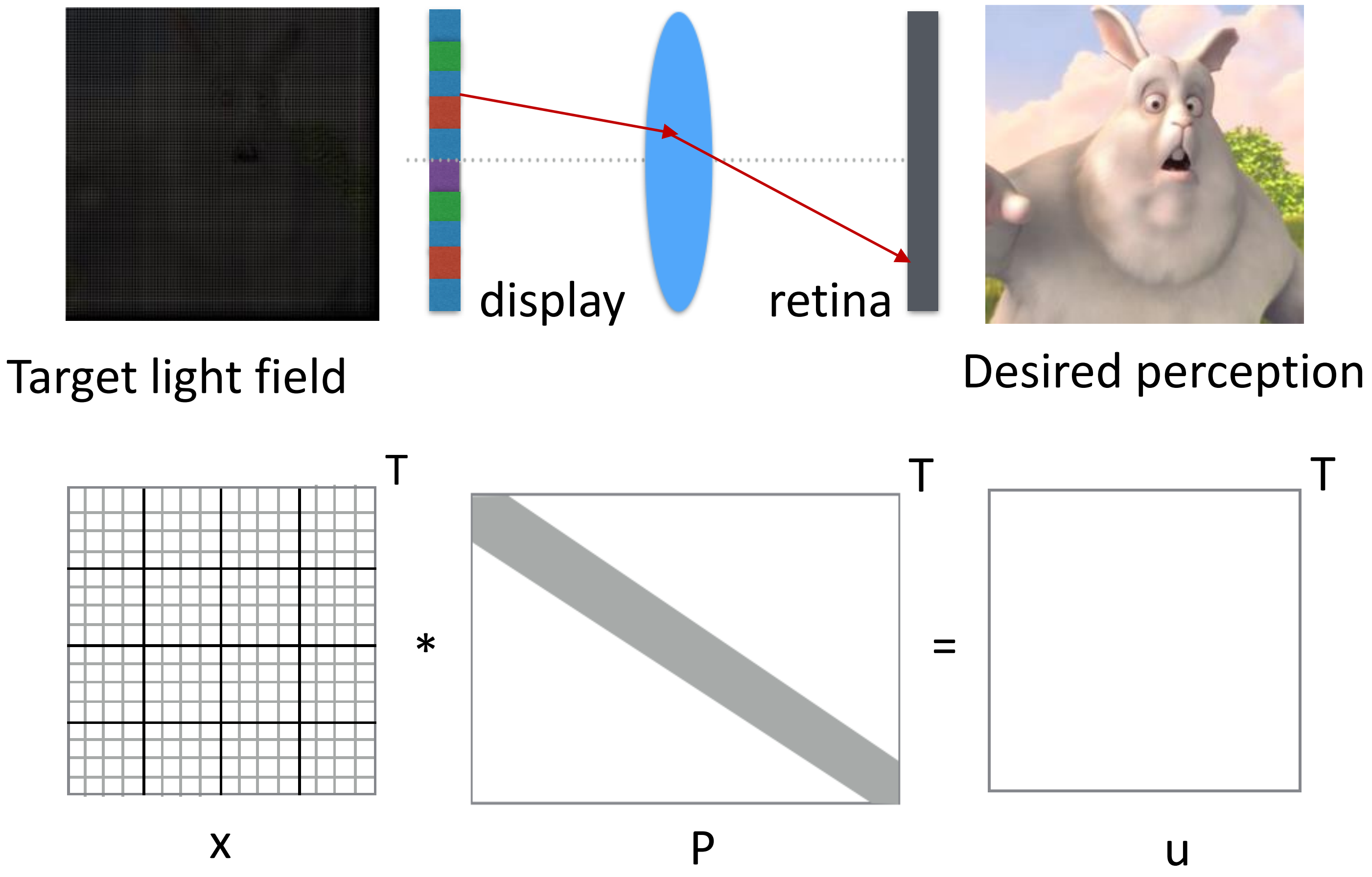
[Huang and Wetzstein, SIGGRAPH 2014]



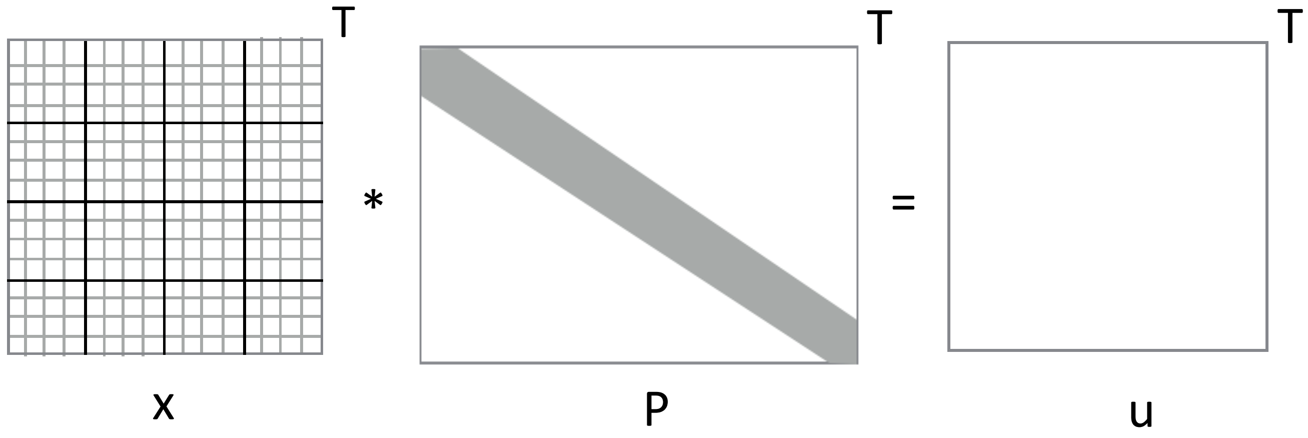
In this Paper…

- Analyze the computational kernels
- Accelerate using FPGAs
- Propose several optimizations

Computational Glass-Free Display



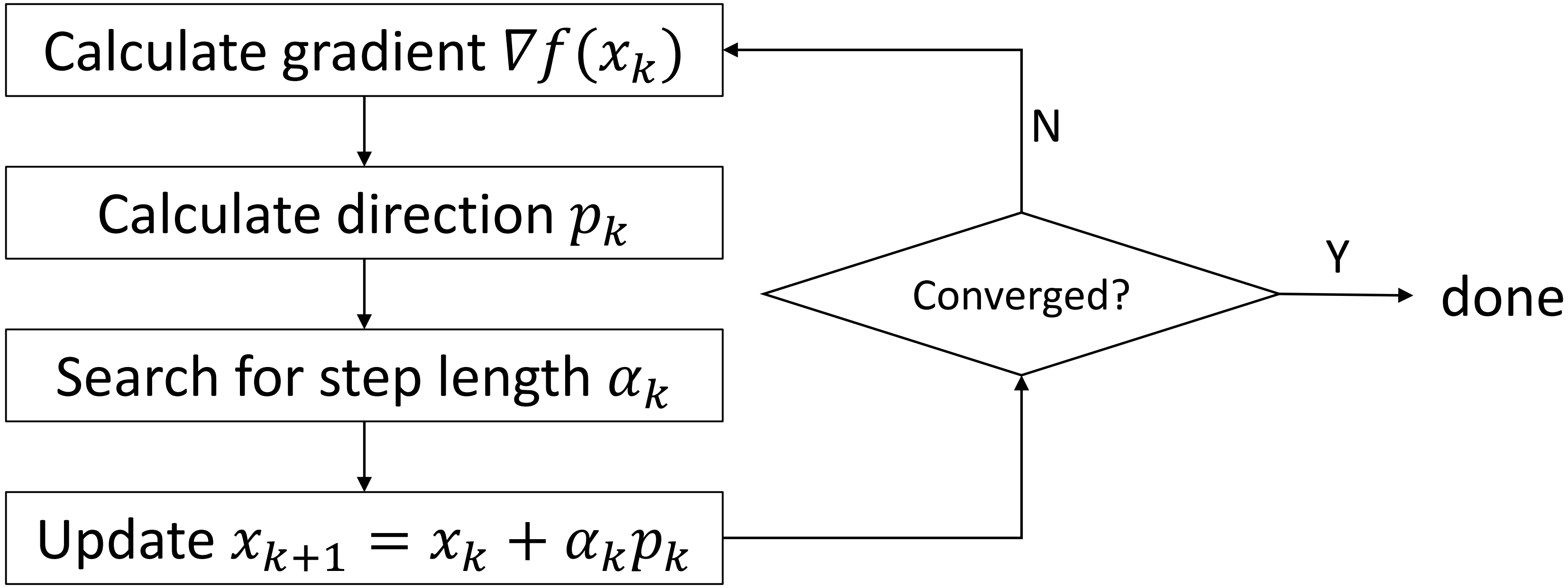
Casting as a Model Fitting Problem



$$\text{minimize} \quad f(x) = \| u - Px \|^2$$
$$\text{subeject to} \quad 0 \leq x \leq 1$$

Background of the L-BFGS Algorithm

- L-BFGS: a widely-used convex optimization algorithm



Background of the L-BFGS Algorithm

- L-BFGS algorithm
  - Input: (history size = m)

    $\overbrace{x_{k-m+1} \quad \cdots \quad x_k}$

    $\nabla f(x_{k-m+1}) \qquad \nabla f(x_k)$

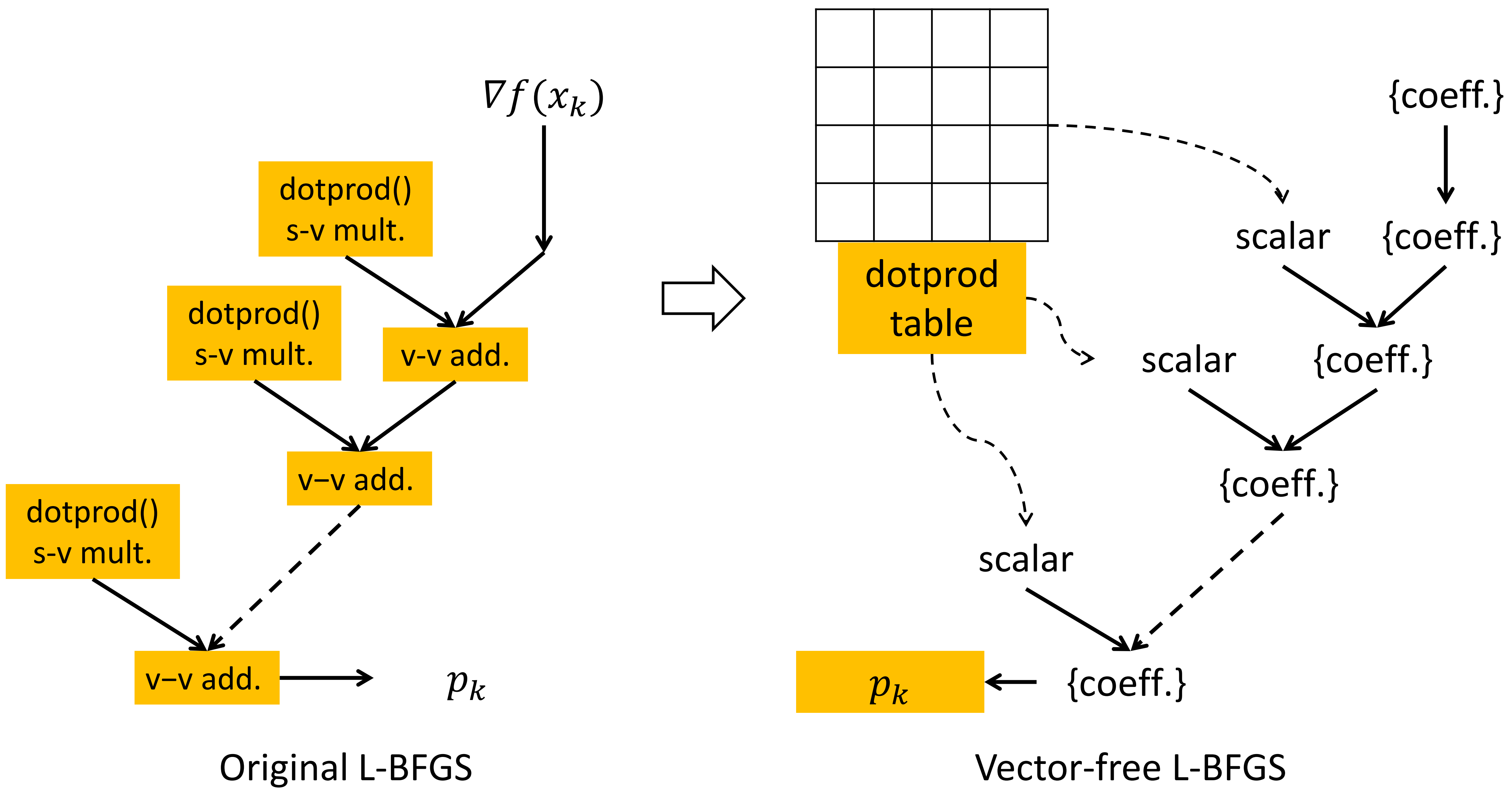
    - $s_j = x_{j+1} - x_j$
    - $y_j = \nabla f(x_{j+1}) - \nabla f(x_j)$
  - Output: direction p_k
- Computational kernels
  - dot prod
  - vector updates

$p_k = -\nabla f(x_k)$
for $i = k-1$ **to** $k-m$ **do**
    [ ] # some work
    $p_k = p_k - \alpha_i y_i$
end for
[ ]
for $i = k-m$ **to** $k-1$ **do**
    [ ] # more work
    $p_k = p_k + (\alpha_i - \beta_i) s_i$
end for
return direction p_k

Vector-free L-BFGS Algorithm

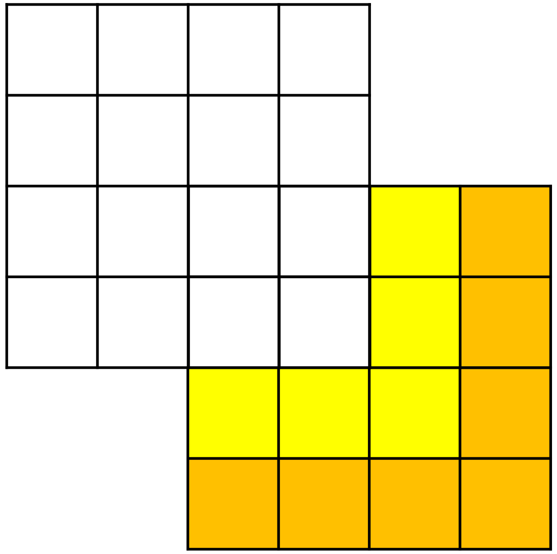
- Original idea
  - [NIPS 2014]
- Observation
  - p_k is a linear combination of the vectors in $\{s_j\}$ and $\{y_j\}$
- Techniques
  - dot prod ⇒ lookup + scalar op.
  - vector update ⇒ coeff. update

$p_k = -\nabla f(x_k)$
for $i = k-1$ **to** $k-m$ **do**
    [ ] # some work
    $p_k = p_k - \alpha_i y_i$
end for
[ ]
for $i = k-m$ **to** $k-1$ **do**
    [ ] # more work
    $p_k = p_k + (\alpha_i - \beta_i) s_i$
end for
return direction p_k


$\nabla f(x_k)$
dotprod()
s-v mult.
dotprod()
s-v mult.
v-v add.
v–v add.
dotprod()
s-v mult.
v–v add.
p_k
Original L-BFGS
dotprod
table
{coeff.}
scalar
{coeff.}
scalar
{coeff.}
{coeff.}
scalar
p_k
{coeff.}
Vector-free L-BFGS

Updating the Dot Product Table

	Scenario	Focus
[NIPS 2014]	Distributed computing using MapReduce	minimize #syncs
Ours	FPGA acceleration with small on-chip BRAM	minimize data transfers



- Similar idea to reduce data transfers
  - dot prod ⇒ lookup + scalar op.
  - vector update => coeff. update

Distributed vs. FPGA-based

	Scenario	Focus	data transfer
[NIPS 2014]	Distributed computing using MapReduce	minimize #syncs	8md
Ours	FPGA acceleration with small on-chip BRAM	minimize data transfers	(4m+4)d

- m: history size (e.g., 10)
- d: image size

Sparse Matrix-Vector Multiplication

$$\text{minimize} \qquad f(x) = \| u - Px \|^2$$

- Size of matrix/vector
  - Sparse matrix P: 16384*490000
  - Variable x: 490000

Sparse Matrix-Vector Multiplication

minimize $\quad f(x) = \| u - Px \|^2$

- Problem: storage of P
- Solution:
  - Sparsity => compressed row storage (CRS)
  - Range of indices => bitwidth reduction
  - #unique values => look-up table (LUT)
    - ~ 810K non-zero entries
    - ~600 unique values

Format	Storage (MB)
flat	32112.64
COO	6.63
CRS	5.24
CRS+LUT	2.90

Sparse Matrix-Vector Multiplication

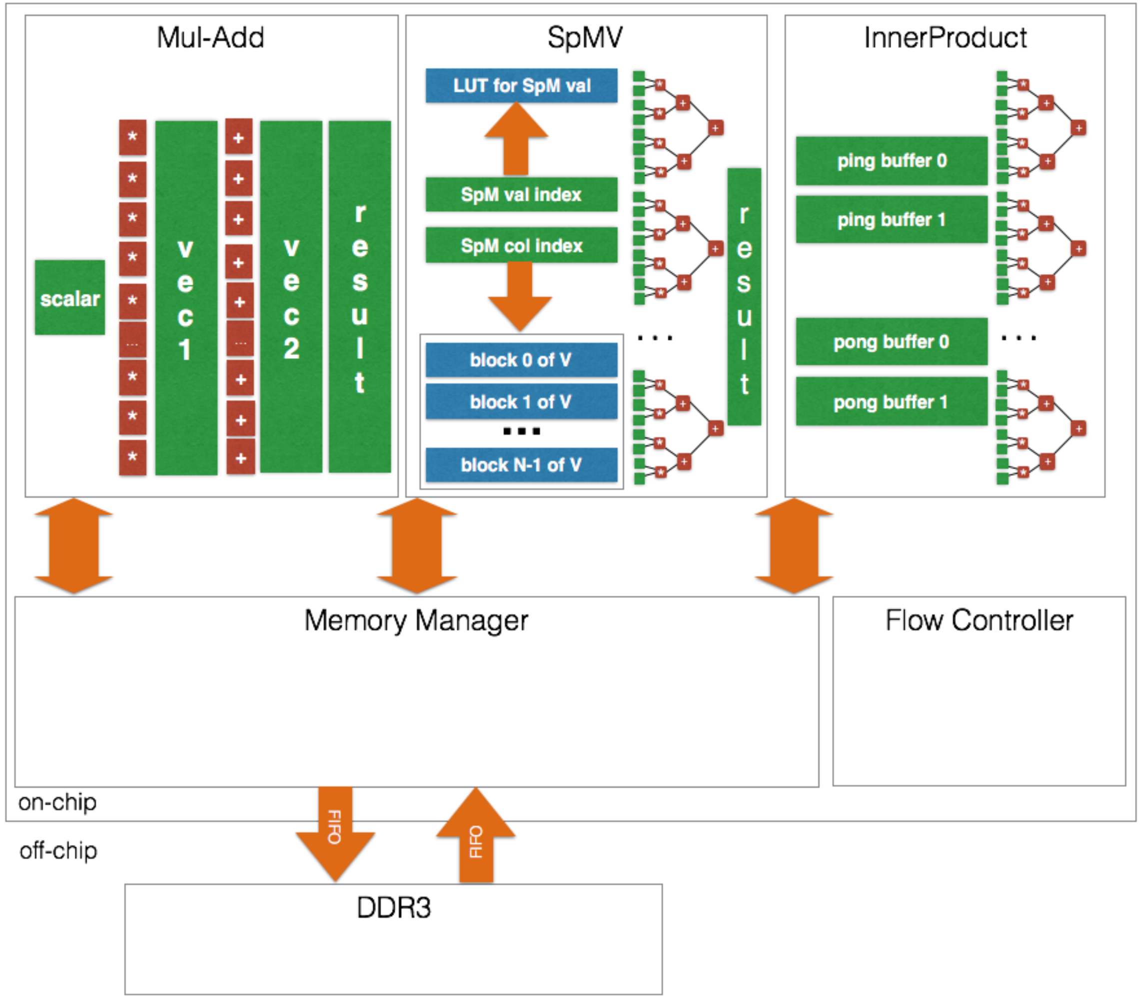
minimize $\quad f(x) =$

- Problem: partitioni
- "Solution":
  - Matrix P is irregula
  - => access pattern i
  - => enumerate fact

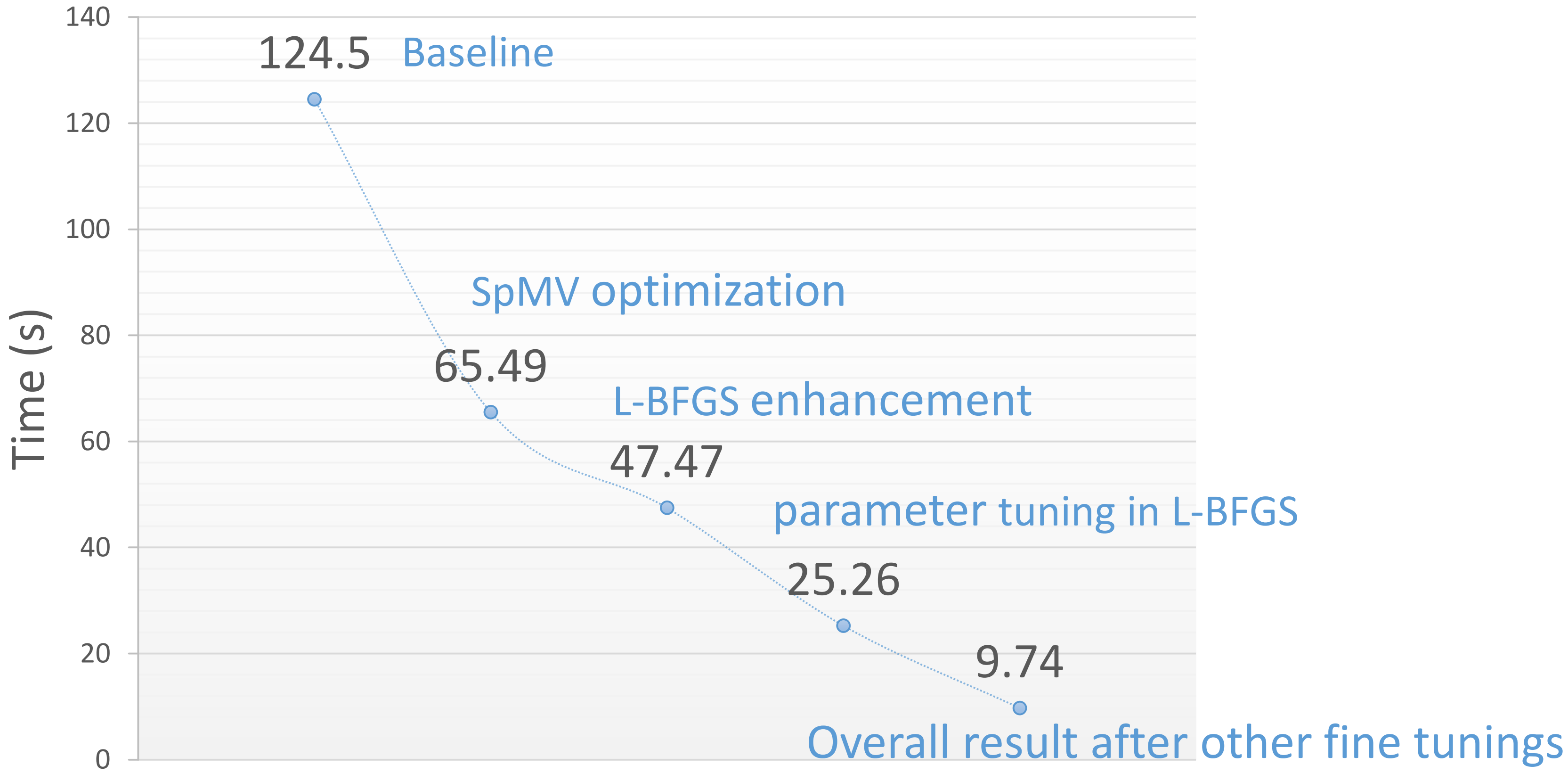
Factor $\boldsymbol{N}$	Method	Min cycle/row	Max cycle/row	Total cycle
980	cyclic	1	1	16384
1225	cyclic	1	1	16384
1250	cyclic	1	2	19840
…	…	…	…	…
1400	block	4	18	188564
1250	block	5	18	193276
…	…	…	…	…
1	N/A	37	54	816272

Overall Design of the Accelerator

- [Li et al, FPGA 15]
- Maximize performance
- Subject to resources



Experimental Evaluation



Runtime Comparison

+: 12.78X Speedup

Conclusions

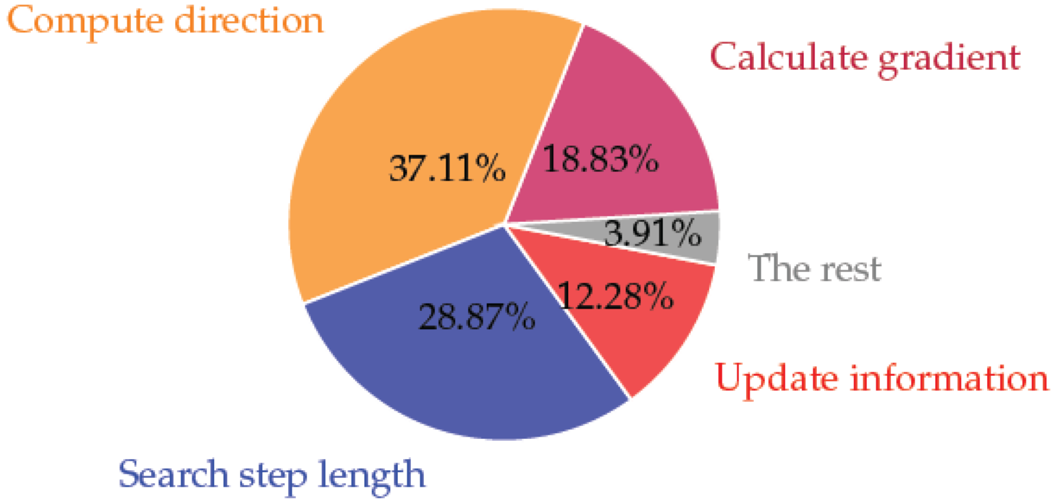
- Summary
  - Bandwidth-friendly L-BFGS algorithm
  - Application-specific sparse matrix compression
  - Memory partitioning for non-affine access

- Future work
  - Possibility of real-time processing
  - Construct transformation matrix by eye-ball tracking
  - A demonstrative system

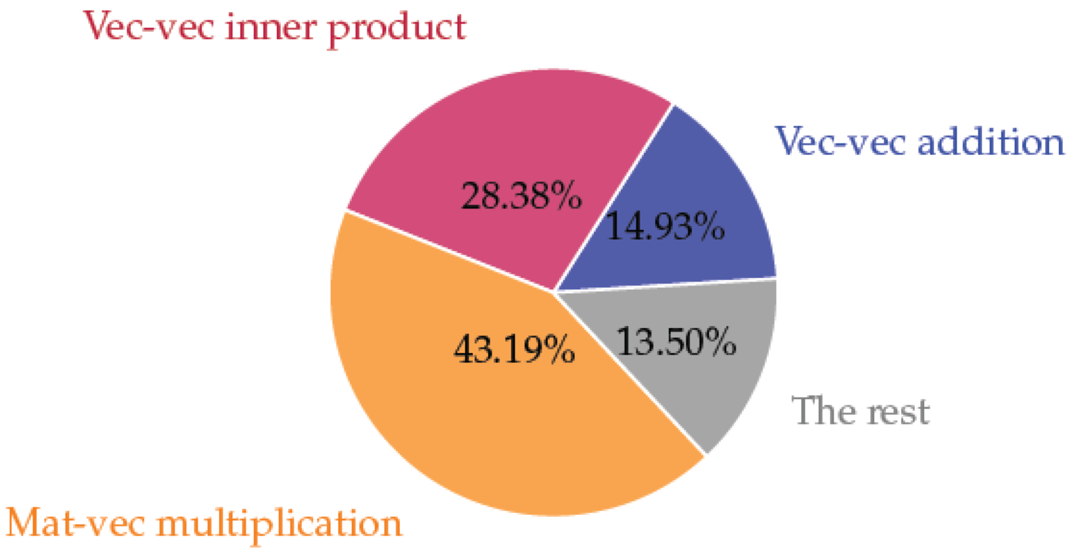
Questions?

Runtime Profiling of a 2-min L-BFGS

per procedure

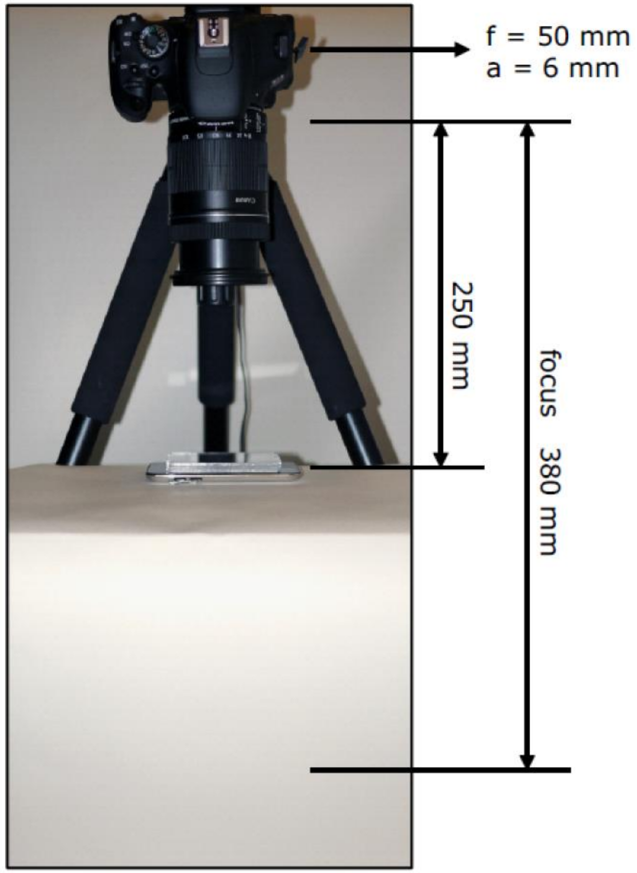


per operation



	LUT	FF	BRAM	DSP
SpMV	8290	6038	1058	10
InnProd	21722	26372	39	0
Mul-Add	27834	40959	169	0
Total	57846	73369	1266	10
Available	303600	607200	2060	2800
Utilization(%)	19	12	61	$\sim$0

Table 5: Resource Utilizations of each Component



(a)experiment setup

source:[Huang and Wetzstein, SIGGRAPH 2014]