Parallel and sequential optical computing

Damien Woods

Reseach Group on Natural Computing
Department of Computer Science and
Artificial Intelligence
University of Seville
Spain

Thomas J. Naughton

Department of Computer Science
National University of Ireland, Maynooth
Ireland

University of Oulu
RFMedia Laboratory
Oulu Southern Institute
Ylivieska, Finland

OSC 2008

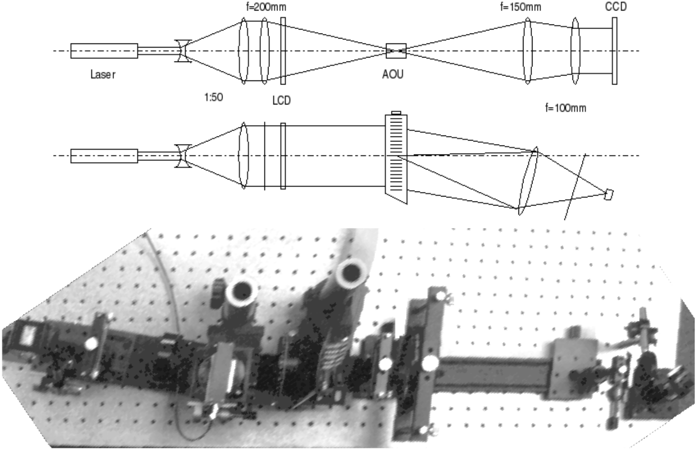
Acknowledgement

- Research Group on Natural Computing, Seville, Spain
- Junta de Andalucía grant TIC-581
- Marie Curie Fellowship through the European Commission Framework Programme 6

Motivations

- Optical computers been around for some time
- Speed: parallelism via 2D complex-valued functions
- No inherent noise during transmission
- Optical pathways can be placed arbitrarily close together
- Photons do not need a conductor to be transmitted (free space propagation)
- High interconnection densities are possible
- Optical pathways can be switched at arbitrary data rates
- Energy efficient (no heat and no additional energy costs for cooling down processors)

Motivations



T. Naughton, et al. Opt. Eng., vol. 38, pp. 1170-1177, 1999.

- Applications: matrix-vector multipliers, image processing
  - e.g. noise removal, edge enhancement, pattern recognition via correlation, numerical computations

Motivations

- Computational complexity of optical computers has received relatively little attention in comparison to the resources devoted to the designs, implementations and algorithms for physical optical computers
- This trend is in contrast to many other models, e.g. quantum computing, DNA computing, membrane computing
- Our work is based on a general optical model inspired by classical Fourier optics (Naughton, 2000)

Continuous space machine definition

- Images are the basic data units
- A *(complex-valued) image* is a function

$$f : [0,1) \times [0,1) \to \mathbb{C}$$

where $[0,1)$ is the half-open real unit interval

Continuous space machine definition

An address is a pair $(\xi, \eta) \in \mathbb{N}^+ \times \mathbb{N}^+$

continuous space machine

A CSM is a quintuple $M = (\mathfrak{E}, L, I, P, O)$, where

- $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ is the address encoding function
- $L = ((s_\xi, s_\eta), (a_\xi, a_\eta), (b_\xi, b_\eta))$ are the addresses: sta, a, b; $a \neq b$
- I and O are finite sets of input and output addresses, respectively
- $P = \{(\zeta_1, p_{1_\xi}, p_{1_\eta}), \ldots, (\zeta_r, p_{r_\xi}, p_{r_\eta})\}$ are the r programming symbols ζ_j and their addresses where $\zeta_j \in (\{h, v, \ldots, hlt\} \cup \mathcal{N}) \subset \mathcal{I}$

configuration

A *configuration* of M is a pair $\langle c, e \rangle$ where

- c is an address called the control
- e is a list of M's images

CSM operations

$\boxed{\text{h}}$	: horizontal 1D Fourier transform on in *a*
$\boxed{\text{v}}$	: vertical 1D Fourier transform on image in *a*
$\boxed{*}$	: complex conjugate of image in *a*
$\boxed{\cdot}$	: pointwise multiplication of *a* and *b*
$\boxed{+}$	: pointwise complex addition of *a* and *b*
$\boxed{\rho}\boxed{z_{\mathrm{l}}}\boxed{z_{\mathrm{u}}}$	: $z_{\mathrm{l}}, z_{\mathrm{u}} \in \mathcal{I}$; filter *a* by amplitude using z_{l} and z_{u} as lower and upper amplitude threshold images
$\boxed{\text{st}}\boxed{\xi_1}\boxed{\xi_2}\boxed{\eta_1}\boxed{\eta_2}$	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy the image in *a* into the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2)
$\boxed{\text{ld}}\boxed{\xi_1}\boxed{\xi_2}\boxed{\eta_1}\boxed{\eta_2}$	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy into *a* the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2)
$\boxed{\text{br}}\boxed{\xi}\boxed{\eta}$	: $\xi, \eta \in \mathbb{N}$; branch to the image at address (ξ, η)
$\boxed{\text{hlt}}$	: halt

CSM programming language

$\mathrm{h}(i_1;i_2)$	: replace image at i_2 with horizontal 1D FT of i_1
$\mathrm{v}(i_1;i_2)$	: replace image at i_2 with vertical 1D FT of i_1
$*(i_1;i_2)$	: replace image at i_2 with complex conjugate of i_1
$\cdot(i_1,i_2;i_3)$	: pointwise multiplication of i_1 and i_2, result in i_3
$+(i_1,i_2;i_3)$	: pointwise addition of i_1 and i_2, result at i_3
$\rho(i_1,z_\mathrm{l},z_\mathrm{u};i_2)$	: filter i_1 by amplitude using z_l, z_u as lower & upper amplitude threshold images

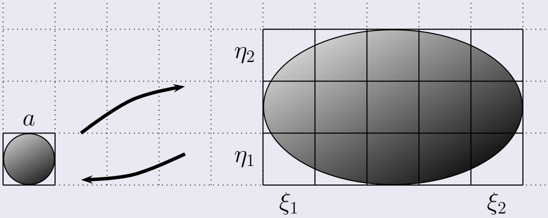
$[\xi_1',\xi_2',\eta_1',\eta_2'] \leftarrow [\xi_1,\xi_2,\eta_1,\eta_2]$: copy the rectangle of images with
- bottom-left address (ξ_1,η_1) & top-right address (ξ_2,η_2)

to the rectangle with
- bottom-left address (ξ_1',η_1') top-right address (ξ_2',η_2')

There are also **if/else** and **while** control flow instructions with binary symbol image conditions.

Example

Copying images



$$[\xi_1, \xi_2, \eta_1, \eta_2] \leftarrow a$$

$$a \leftarrow [\xi_1, \xi_2, \eta_1, \eta_2]$$

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x,y),(\Phi,\Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x,y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x,y)$ in the computation of M is replaced with $S(f(x,y),(\Phi,\Psi))$ then M computes correctly on all inputs.

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.

CSM complexity measures

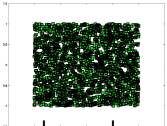
Recall that $f(x,y) = |f(x,y)|\mathrm{e}^{\mathrm{i}\arg f(x,y)}$. Let $A : \mathcal{I} \times \mathbb{N}^+ \to \mathcal{I}$,

$$A(f(x,y),\mu) = \left\lfloor |f(x,y)|\mu + \frac{1}{2} \right\rfloor \frac{1}{\mu}\,\mathrm{e}^{\mathrm{i}\arg f(x,y)}$$

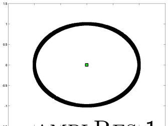
The value μ is the cardinality of the set of discrete nonzero amplitude values that each complex value in $A(f,\mu)$ can take, per half-open unit interval of amplitude.

AMPLRES

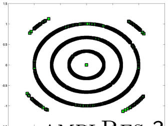
The minimum μ such that if each image $f(x,y)$ in the computation of M is replaced by $A(f(x,y),\mu)$ then M computes correctly on all inputs.



random values



AMPLRES 1



AMPLRES 3

CSM complexity measures

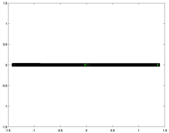
Let $P : \mathcal{I} \times \mathbb{N}^+ \rightarrow \mathcal{I}$,

$$P(f(x,y),\mu) = |f(x,y)|\mathrm{e}^{\mathrm{i}\left\lfloor \arg(f(x,y))\frac{\mu}{2\pi}+\frac{1}{2}\right\rfloor\frac{2\pi}{\mu}}$$

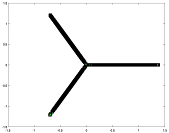
The value μ is the cardinality of the set of discrete phase values that each complex value in $P(f,\mu)$ can take.

PHASERES

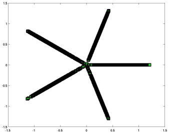
The minimum μ such that if each image $f(x,y)$ in the computation of M is replaced by $P(f(x,y),\mu)$ then M computes correctly on all inputs.



PHASERES 2



PHASERES 3



PHASERES 5

CSM complexity measures

DYRANGE

The ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.

FREQ

The minimum optical frequency such that M computes correctly on all inputs.

SPACE

The product of all of M's complexity measures except TIME.

We have defined complexity of computations, we extend this to complexity of configurations and images in a straightforward way.

CSM complexity measures

DYRANGE

The ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.

FREQ

The minimum optical frequency such that M computes correctly on all inputs.

SPACE

The product of all of M's complexity measures except TIME.

We have defined complexity of computations, we extend this to complexity of configurations and images in a straightforward way.

$\mathcal{C}_2$-CSM

Motivated by a desire to apply standard complexity theory tools to the model, we define a restricted class of CSM.

$\mathcal{C}_2$-CSM

- AMPLRES and PHASERES have constant value of 2
- at TIME t each of GRID, SPATIALRES, DYRANGE is $O(2^t)$
- h and v compute horizontal and vertical DFT respectively (SPACE is redefined to be the product of all complexity measures except TIME and FREQ)
- the address encoding function $\mathfrak{E} : \mathbb{N} \rightarrow \mathcal{N}$ is decidable by a logspace Turing machine (given a *reasonable* binary word representation of the set of addresses $\mathcal{N}$)

2005: lower and upper bounds on $\mathcal{C}_2$-CSM power

The $\mathcal{C}_2$-CSM verifies the parallel computation thesis

$\Leftrightarrow$ $\mathcal{C}_2$-CSM TIME is (polynomially) equivalent to sequential space

$\Leftrightarrow$ $\mathcal{C}_2\text{-CSM–TIME}(S^{O(1)}(n)) = \mathrm{NSPACE}(S^{O(1)}(n))$

For example, $\mathcal{C}_2\text{-CSM–TIME}(n^{O(1)}) = \mathrm{PSPACE}$
For example, $\mathcal{C}_2$-CSMs solve NP-complete problems polynomial time, but (naturally) use exponential space.

Poly SPACE, polylog TIME $\mathcal{C}_2$-CSMs accept exactly NC
i.e. $\mathcal{C}_2\text{-CSM–SPACE, TIME}(n^{O(1)}, \log^{O(1)} n) = \mathrm{NC}$

These characterisations are robust wrt variations in the $\mathcal{C}_2$-CSM definition

2005: lower and upper bounds on $\mathcal{C}_2$-CSM power

The $\mathcal{C}_2$-CSM verifies the parallel computation thesis

$\Leftrightarrow$ $\mathcal{C}_2$-CSM TIME is (polynomially) equivalent to sequential space

$\Leftrightarrow$ $\mathcal{C}_2\text{-CSM–TIME}(S^{O(1)}(n)) = \text{NSPACE}(S^{O(1)}(n))$

For example, $\mathcal{C}_2\text{-CSM–TIME}(n^{O(1)}) = \text{PSPACE}$
For example, $\mathcal{C}_2$-CSMs solve NP-complete problems polynomial time, but (naturally) use exponential space.

Poly SPACE, polylog TIME $\mathcal{C}_2$-CSMs accept exactly NC
i.e. $\mathcal{C}_2\text{-CSM–SPACE, TIME}(n^{O(1)}, \log^{O(1)} n) = \text{NC}$

These characterisations are robust wrt variations in the $\mathcal{C}_2$-CSM definition

2005: lower and upper bounds on $\mathcal{C}_2$-CSM power

The $\mathcal{C}_2$-CSM verifies the parallel computation thesis

$\Leftrightarrow$ $\mathcal{C}_2$-CSM TIME is (polynomially) equivalent to sequential space

$\Leftrightarrow$ $\mathcal{C}_2\text{-CSM–TIME}(S^{O(1)}(n)) = \text{NSPACE}(S^{O(1)}(n))$

For example, $\mathcal{C}_2\text{-CSM–TIME}(n^{O(1)}) = \text{PSPACE}$
For example, $\mathcal{C}_2$-CSMs solve NP-complete problems polynomial time, but (naturally) use exponential space.

Poly SPACE, polylog TIME $\mathcal{C}_2$-CSMs accept exactly NC
i.e. $\mathcal{C}_2\text{-CSM–SPACE, TIME}(n^{O(1)}, \log^{O(1)} n) = \text{NC}$

These characterisations are robust wrt variations in the $\mathcal{C}_2$-CSM definition

- We already know that:
- PSPACE $= \mathcal{C}_2$-CSM poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$ pixels

- Backed up by existing intuition through many, many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- PSPACE $= \mathcal{C}_2$-CSM poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$ pixels

- Backed up by existing intuition through many, many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2\text{-CSM poly-}\textsc{time}\text{, exp-}\textsc{spatialRes}$
- Result holds for constant $O(1)$ usage of the other resources: $\textsc{dyRange}$, $\textsc{grid}$, $\textsc{amplRes}$, $\textsc{phaseRes}$

parallelism $\approx$ pixels

- Backed up by existing intuition through many, many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2$-CSM poly-$\textsc{time}$, exp-$\textsc{spatialRes}$
- Result holds for constant $O(1)$ usage of the other resources: $\textsc{dyRange}$, $\textsc{grid}$, $\textsc{amplRes}$, $\textsc{phaseRes}$

parallelism $\approx$ pixels

- Backed up by existing intuition through many, many examples of optical algorithms

Parallelism without pixels?

- What if we fix the number of pixels?

i.e. $O(1)$ SPATIALRES

- Our previous highly parallel algorithms don't work
- Have we crippled the system?
- No!

Constant number of pixels

Theorem

PSPACE *is characterised by* $\mathcal{C}_2$-CSMs *that are restricted to use polynomial* TIME $T = O(n^k)$, SPATIALRES $O(1)$, GRID $O(1)$, *and generalised to use* AMPLRES $O(2^{2^T})$, DYRANGE $O(2^{2^T})$.

- Proof (upperbound). Extend previous upperbound, swapping the roles of SPATIALRES and the other resources.
- Proof (lowerbound). Via simulation of RAM($\times, +, \leftarrow$). Such RAMs are known to characterise PSPACE in polynomial time.

Constant number of pixels

Theorem

PSPACE *is characterised by* $\mathcal{C}_2$-CSMs *that are restricted to use polynomial* TIME $T = O(n^k)$, SPATIALRES $O(1)$, GRID $O(1)$, *and generalised to use* AMPLRES $O(2^{2^T})$, DYRANGE $O(2^{2^T})$.

- We can get "high parallelism" with a fixed number of pixels!
- However, not a realistic way to do optical computing: using large AMPLRES and DYRANGE is more expensive and unrealistic than large SPATIALRES and/or GRID
- Using multiplication, rather than pixels
- Intuition — there are at least two ways to compute quickly in optics: use pixels or generate large numbers
- So what happens if we dissallow (such unrealistic) multiplication?

What happens if we remove the multiplication operation?

Theorem

$\mathcal{C}_2$-CSMs without multiplication, that compute in polynomial TIME, *polynomial* GRID $O(n^k)$, *and constant* SPATIALRES $O(1)$, *characterise P.*

Theorem

$\mathcal{C}_2$-CSMs without multiplication, that compute in polynomial TIME, *constant* GRID $O(1)$, *polynomial* SPATIALRES $O(n^k)$, *characterise P.*

- Significant reduction in power
- These results are general in the sense that the other resources are arbitrary (i.e. unrestricted GRID, DYRANGE, PHASERES, AMPLRES)

Definition (Needle in haystack problem)

Let $L = \{w : w \in 0^*10^*\}$. Let $w \in L$ be written as $w = w_0 w_1 \dots w_{n-1}$ where $w_i \in \{0, 1\}$. Given such a w, the needle in haystack problem asks what is the index of the symbol 1 in w. The solution to the needle in haystack problem for a given w is the index i, expressed in binary, where $w_i = 1$.

- Grover's quantum algorithm: $O(\sqrt{n})$
- CSM algorithm: $O(\log(n))$

Searching in log time

Represent $w = 0^{i-1}10^{|w|-i}$ as a binary valued image, with value 1 at horizontal position i, and value 0 elsewhere

Repeat $\log_2 n$ times

1. FT, square, FT left half of image
2. If centre is nonzero
   - append 0 to address
   - discard right half of image
3. Else
   - append 1 to address
   - discard left half of image

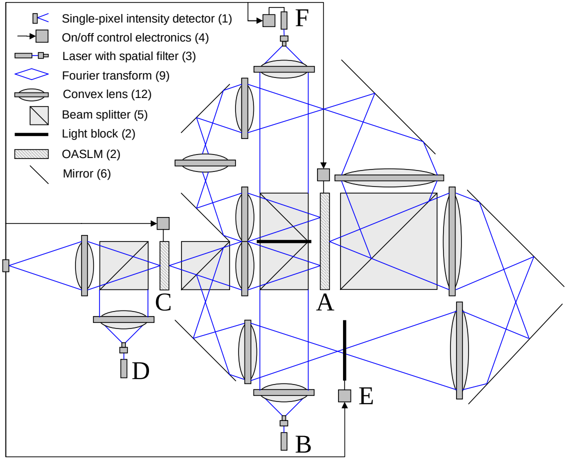
CSM algorithm to search for a single 1 in a string of 0s.

	i1	i2			f_0	f_1			sta					a	b	c	d	e									
99									br	0	3																
w	ld	e	st	de	br	0	$\widehat{\text{d}}$																				
3	ld	i2	st	e	ld	f_0	st	c	br	0	w																
2	h	v	st	b	$*$	$\cdot$	h	v	br	8	$\widehat{\text{a}}$																
1	ld	i1	st	ab	br	0	2		ld	i1	st	ab	st	i1	ld	f_0	st	b	ld	bc	st	c	br	0	w		
0	ld	c	hlt						ld	i1	st	ab	ld	b	st	i1	ld	f_1	st	b	ld	bc	st	c	br	0	w
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 . . .										

Fig. 9. CSM to perform a $\Theta(\log_2 n)$ search on an unordered binary list.

35

Optical apparatus to search for a single 1 in a string of 0s.



Conclusions

- Two ways in which we get huge parallelism from optics:
- Characterise PSPACE in poly TIME, but exp SPATIALRES
- Char. PSPACE in poly TIME, but exp AMPLRES & DYRANGE
- Remove multiplication $\Rightarrow$ characterise P
- Corollaries: characterisations of NC via optical machines that run in polylog TIME & polynomial SPACE
- So we can take existing, fast parallel algorithms and compile to fast parallel optical algorithms
- log time searching algorithm & implementation design

Conclusions

- NC: problems solved in polylog time, and poly space
- NC $\subseteq$ P,
- Rather than focus on (presumed) hard problems, perhaps the optical computing community can get more out of optical computers by finding NC problems that are well-suited to optical architectures